Tidal Migrations Layered Discovery Guide

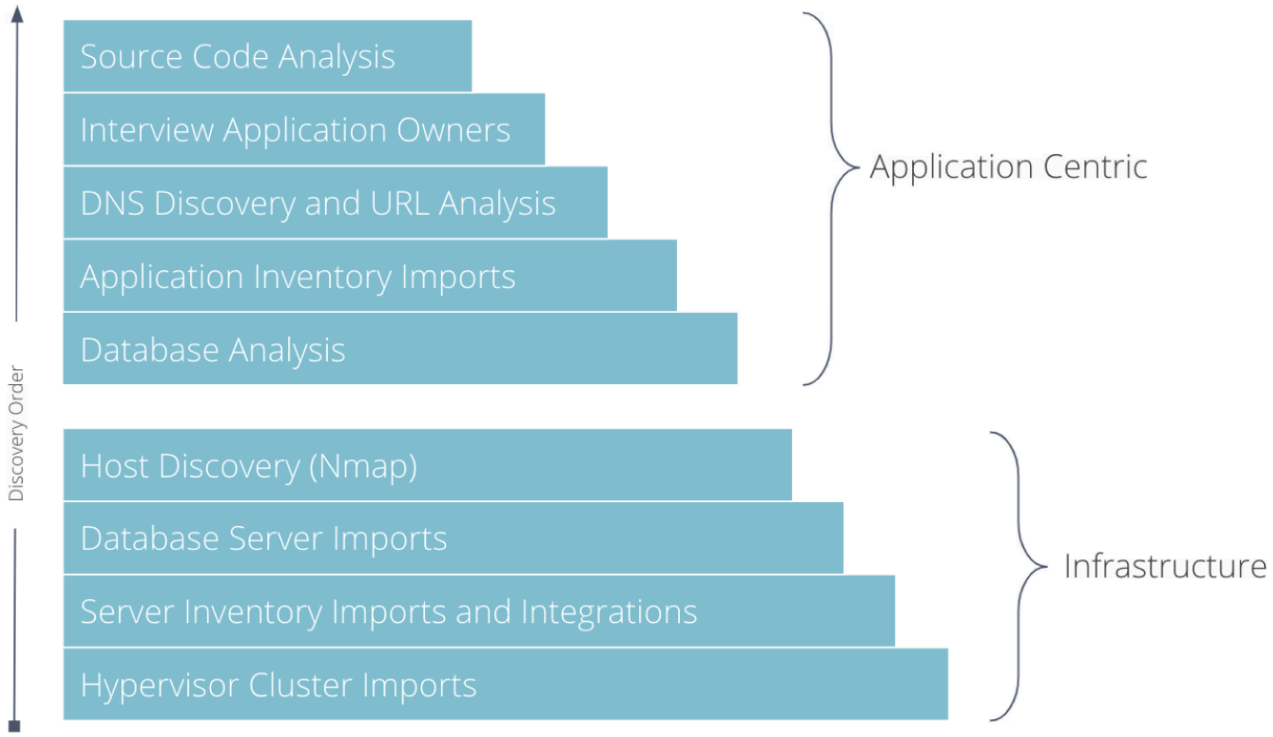
tidal migrations

Contents

Layering discovery techniques

This book will cover all the discovery steps you will need to perform for your cloud migration project.

Cloud migration is the process of moving your data, applications, and other elements to the cloud. However, the path to the cloud can be long and painful without proper planning and execution. By following Tidal Migration's six discovery layering techniques, you will be migrating to the cloud with ease!



Tidal Migrations' layered approach to application discovery, for cloud migration.

Six Steps To Discovery Bliss

1) Import Your Spreadsheets

If your already have some data collected in spreadsheets, the first step to begin your cloud migration project is importing a spreadsheet of Virtualization Clusters, Servers, Databases Instances and Applications. Tidal Migration's importer will guide you through mapping your columns to our fields, create your own fields and even make associations between dependencies if you have captured these.

NB: See additional ways on importing your applications and servers in the API docs.

---

2) Integrate Your Hypervisors

Once you have imported your data, you can begin to synchronize your inventories via `tidal sync servers`. Tidal sync supports many server inventory management tools such as VMWare and HyperV with more

possible via scripting (ask us[1]).

If you have VM Ware's vSphere, `tidal sync vsphere` will handle everying with just read-only credentials required.

Scheduling your sync:

It is useful to setup a cron job or Windows Scheduled Task for this process, and we recommend synchronizing your inventories no more than once per day.

This will keep your resource inventory up to date and show you usage trends over time in the *Analyze* feature.

---

3) Aggregate Server Usage Statistics

Tidal Migrations provides you with a simple and effective way to gather machine statistics[2] (RAM, Storage, CPU allocations and usage) from Windows and UNIX/Linux server environmens. In Windows, we use WinRM to Invoke-Command across your servers, and for *NIX we leverage ansible. Both of these approaches output JSON which is securely sent to your Tidal Migrations instance using the tidal command.

Checkout this guide for a quick and clean approach to gathering server usage statistics. See the machine_stats[3] repository for more implementation details.

NB: Feel free to fork the repo and modify to suit your needs, or to show your security team and give them comfort. This extensibility and transparency is core to our approach.

---

4) Fingerprint Web Applications

a.) The initial step in your cloud journey is discovering what you have. It can be hard to remain informed about all the domains and applications hosted in your environment, which is why we created the `tidal discover` command. With your customized *Discovery Plan* you can obtain both private and public domains within your datacentres in under 60 seconds.

This tidal-tools guide contains examples for creating your own *Discovery Plan* to scan multiple networks and DNS services.

b.) With a list of domains in hand, the next step is to *analyze* the applications hosted on these domains.

`tidal analyze web` will fingerprint the technology on both your internet sites and intranet applications behind your firewall in seconds, *without needing to install agents.* Whether you have 1 or 1 million end points, Tidal Tools centralizes the data gathered into our platform for you to analyze further and plan with, simplifying your application centric cloud migration.

For detailed information and steps on analyzing your domains, be sure to checkout this guide.

---

5) Analyze Your Databases

Analyze all of your databases in minutes and *measure* the migration difficulty.

After running `tidal analyze db your_config.yml` against your databases (see guide), you will understand which database features in your Oracle,SQL Server, MySQL, or PostgreSQL installations make it difficult to adopt cloud-native database offerings and also identify which applications are connecting to your databases.

With over 100 unique characteristics considered, comparisons are made with the data platforms available in the cloud(s) you are migrating to which provide you with *data-driven insights* for planning your cloud migration.

[1] https://tidalmigrations.com/contact
[2] https://github.com/tidalmigrations/machine_stats
[3] https://github.com/tidalmigrations/machine_stats

Follow the steps in the guide, and you will be able to quantify the difficulty in migrating your database from Oracle to PostreSQL; or from SQL Server to AWS Aurora etc.

---

6) Analyze Your Source Code

Finally, to find the applications which will migrate more easily to cloud-native technologies you can analyze your source code and rank your applications by *Cloud Readiness*.

Doing this for each of your custom applications which have a *Transition Type* of Refactor or Replatform will give you the data needed to prioritize your application migrations. To analyze your source code, you need the `Application ID`, to be logged in with tidal-tools and a copy of the source code checked out.

You can find the `Application ID` in the URL bar when looking at an application. e.g. if I'm loking at an application in Tidal Migrations, the URL will show `https://demo2.tidalmg.com/#/apps/111` Here, `111` is my `Application ID`.

I can now analyze the source code with:

```
cd /path/to/source-code
tidal analyze code --app-id 111 .
```

To find additional information about this feature, visit the guide on analyzing your source code.

The remaining will now go through the discvory process step by step. Starting with installing Tidal Tools.

Getting started with Tidal Tools

Note

This page describes general installation and configuration instructions for Tidal Tools. If you are looking for specific AWS usage instructions please refer to Getting Started with Tidal Tools on AWS guide.

Here we outline how to get started working with Tidal Tools via the command line.

Downloading & Installing

Tidal Tools is a cross platform CLI utility that you can use to discover and analyze your applications. Easily install Tidal Tools[4] on your operating system.

Dependencies

If you plan to use the `tidal analyze code` command to analyze your source code or `tidal analyze db` to analyze your databases, you will need to install Docker Community Edition. To install Docker on your system you can check Docker's documentation directly[5]. Once installed you can verify it is installed and working correctly with the command `tidal doctor`.

Docker for Windows supports both Linux containers and Windows containers, however **Tidal Tools works when your Docker installation is set to use Linux containers**. Set up Docker for Windows to use Linux containers.

Using Tidal Tools

To get started, from a new terminal or Powershell prompt, simply run:

```
tidal
```

To see what you can do with the tidal checkout some of our other articles about creating sync jobs or analyzing your applications via their URLs.

Connecting to the API

Once you have Tidal Tools installed you need to configure access to the API. Register for a free account[6] with Tidal Migrations to get the connection details.

There are several ways to authenticate with the Tidal Migrations API, we recommend the first one, `tidal login`, because your password is never persisted to disk.

[4] https://get.tidal.sh
[5] https://docs.docker.com/install/
[6] https://get.tidalmg.com/

Tidal Login command

To authenticate with the API type `tidal login` and follow the prompts. This should provide and store an access token for you that gives you access for 8 hours. Once it is expired, the user must login again and obtain another token.

We recommend that you utilise this command because it doesn't store your password. If you must store the password, you can authenticate using the methods below.

Alternative authentication methods

Configuration file

Alternatively, you can use the `tidal config` command to manually set your credentials.

For example, you can set your username, password and URL with the three following commands:

```
tidal config set tidal.email [set your email]
tidal config set tidal.password [set your password]
tidal config set tidal.url https://demo.tidalmg.com
```

Your credentials will be stored in a configuration file such as:

```
tidal:
  email: my_user_name_here
  password: my_secure_password_here
  url: https://my_instance_name_here.tidalmg.com
```

On macOS the config file is located: `$HOME/Library/Preferences/tidal/config.yaml`

On Linux systems it is: `$HOME/.config/tidal/config.yaml`

And on Windows it is: `%AppData%\tidal\config.yaml`

Environment Variables

You can specify your credentials as environment variables. If so, these variables need to be set:

- `TIDAL_EMAIL`
- `TIDAL_PASSWORD`
- `TIDAL_URL`

Dynamically

Alternatively you can pass in your credentials on each request as command line arguments using the following flags:

```
--tidal-email youremail@address.here
--tidal-password your_secure_password
--tidal-url https://yoursubdomain.tidalmg.com
```

Testing connectivity and authentication

Once you have set your credentials you can test your connectivity and authentication to the API with the ping command:

```
tidal ping
```

Using a Proxy

If you need to use a proxy, you can specify it with the `https_proxy` variable.

On macOS and Linux you would run:

```
export https_proxy=1.1.1.1:123
```

and on Windows:

```
set https_proxy=1.1.1.1:123
```

Of course replacing `1.1.1.1` with your proxy IP address and `123` with the port number.

Once you have this set, you should be able to run `tidal ping` successfully. Remember that you will need to set this for every new terminal session you start.

Import data from Excel

Tidal Migration's importer will guide you through mapping your columns to our fields, create your own fields and even make associations between dependencies if you have captured these.

You can import your Excel spreadsheets into Tidal Migrations by visiting:

`https://your_subdomain.tidalmg.com/#/discover/import`

Preparing

You should import your data in the following order:

1. Virtualization Clusters
2. Servers
3. Database Instances
4. Applications

This order is especially important if you are planning to import dependencies for your Databases and Applications. Ie. If your Application has a dependency on a server, you need to have that server imported first for the dependency to be imported correctly.

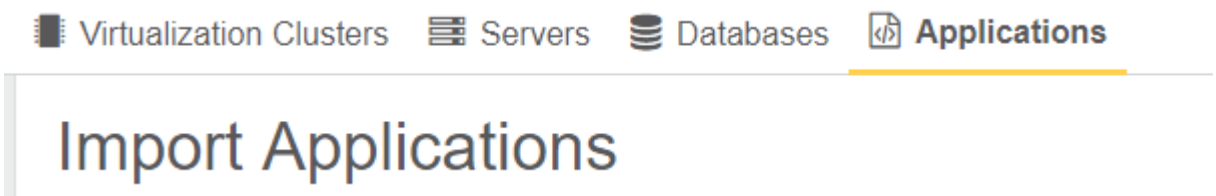
> **Note**
>
> You can always update a dependency after importing too.

You should have one sheet or file per type of record.

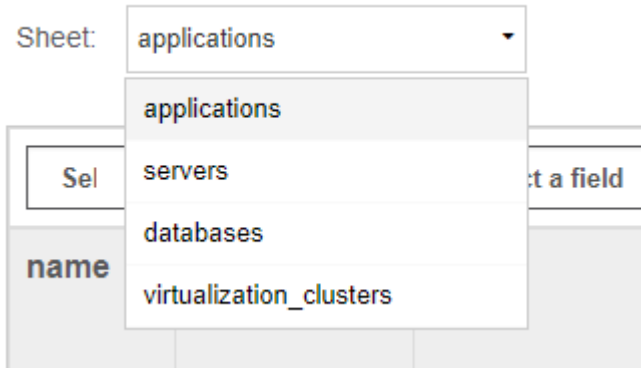
There are several default columns included for each record type, however if you have additional information you can add fields for a record and also track those. If this is the case you should do this before you import your data.

Importing your data

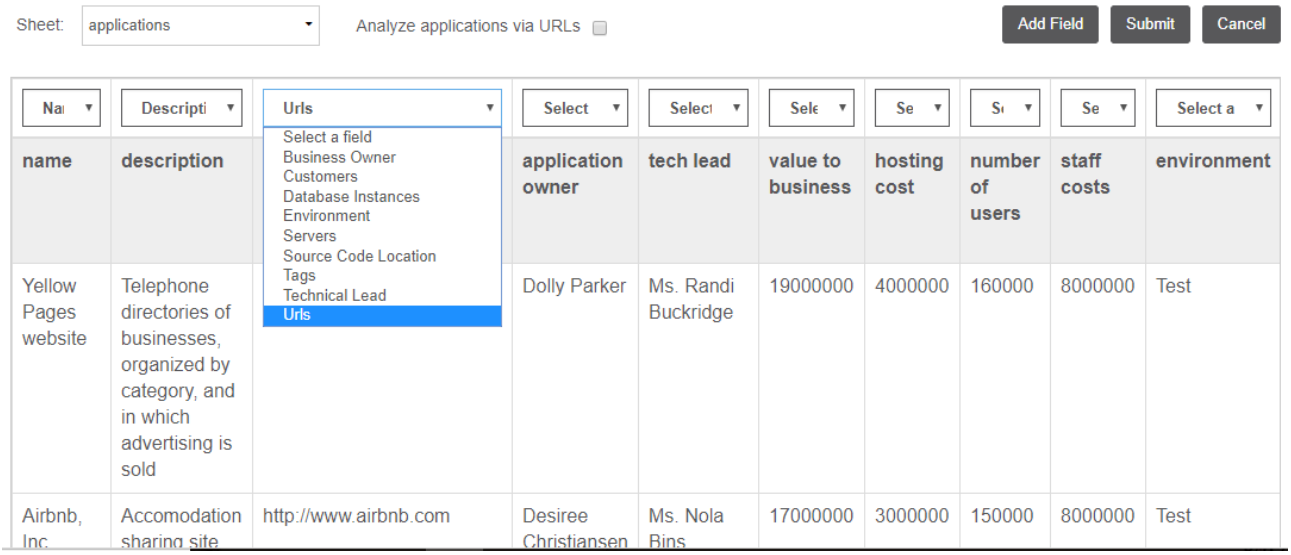
- Select the data type you wish to import.



- Select the sheet in your file that you want to import data from.



- Map the columns of your sheet to the fields from the drop down. If you have a column that is not present as a field and you want to import it, you can add that field.



- Click submit to import the data.

Tip

The Tags option can be used to generate multiple tags and associate them with the record. You can use this track any type of categorical data. Additionally you can search and filter for records based on tags. Multiple tags can be associated based on one column of data, they are parsed and split on any commas (,)

Importing Virtualization Clusters

Tidal Migration uses the following fields to map a virtualization cluster to other resources: *Hypervisor Technology, Servers*. For this field, you must select the property that the column will correspond to, generally this is *Name*.

Importing Servers

Tidal Migrations uses the following fields to map a server to other resources: *FQDN, IP Addresses, Database Instances, Environment, Cluster*. For these fields, you must select the property that the column will correspond to, generally this is *Name*.

Importing Databases

Tidal Migrations uses the following fields to map a database instance to other resources: *Servers and Environment*. For these fields, you must select the property that the column will correspond to, generally this is *Name*.

Importing Applications

Tidal Migrations uses the following fields to map an application to other resources: *Servers, Database Instances, Environment, Customers*. For these fields, you must select the property that the column will correspond to, generally this is *Name*.

Sync with your hypervisors

Getting Started

To get started you will need to make sure you have a copy of Tidal Tools[7] and it is installed.

Supported Versions

vSphere 5.5

NB: When you set the vsphere.server you will likely need to include the default port, 443, in the url. Ex. 192.168.1:443 ### vSphere 6.0

vSphere Login

If you have not logged on to the Tidal API yet, run `tidal login` to do so.

Once you are logged on to the Tidal API, execute the `tidal login vsphere` command

A set of prompts will appear to help you connect to your vSphere account:

1. vSphere Server - Either an IP address or a FQDN that resolves to the correct IP for the ESXi or vCenter instance
2. vSphere Username - A username that has read access to the vCenter or ESXi instance.
3. vSphere Password - Corresponding password for the user.
4. vSphere API Endpoint - The default is `/sdk`, It can be left as is, unless you know your setup does not use the default.
5. Use SSL/TLS? - The default is to use SSL (y), which can be left unless you know you can't use it.
6. Skip HTTPS certificate checks? - The default is to not skip (n), which can be left unless you know they need to be skipped.

After answering the above prompts, you will see the message:

```
Login to vSphere successful. Saving config file...Done!
```

Sync

Syncronizing your vSphere inventory with Tidal Migrations is simple with:

```
tidal sync vsphere
```

```
Loading data..
Transforming data..
Validating JSON...
Valid.
Logging in to server at https://demo.tidalmg.com as info@tidalmigrations.com
Logged in.
Updating via API..
Update successful
```

[7] https://get.tidal.sh

You can easily split this into two commands to allow you to inject other metadata about your inventory, if needed. To do this simply use the get command to get your inventory. Modify it as you need and then you can use the sync command to upload your data.

1. `tidal get vsphere`, will fetch your vSphere inventory and output JSON (check it out!).
2. Modify the data as you like from Standard Input and then send it to Standard Output.
3. `tidal sync servers`, will take that JSON from STDIN and send it to Tidal Migrations.

```
tidal get vsphere | ./modify_script.rb | tidal sync servers
```

Repeat

You can easily set this to run periodically. The integration updates records if they already exist, or creates new records if they don't. Look at setting this command up as a cron job[8] that runs once per day.

Additional Configuration Options

The integration can be configured with several other methods besides `tidal login`. You can use these alternate approaches if they better suite your use case.

The other configuration options available are to set values in the configuration file (either interactively or manually) or set environment variables.

The credentials and configuration settings take precedence in the following order:

1. Environment Variables
2. Configuration file
3. Prompt in the terminal - If neither are present, you will be prompted to enter the required details and they are stored in the default configuration file.

1) Configuration File

Set Interactively

You can set your vSphere credentials with the following commands:

```
tidal config set vsphere.username [your username]
```

```
tidal config set vsphere.password [your password]
```

```
tidal config set vsphere.server 192.168.1.12
```

The set values are stored in the default configuration file.

Set Manually

You can create a file, for example `config.yml`, with content similar to this:

```
vsphere:
  username: my_user_name_here
  password: my_secure_vsphere_password
  server: 1.1.1.1
tidal:
  email: my_user_name_here
  password: my_secure_password_here
  url: https://my_instance_name_here.tidalmg.com
```

That can be used with a command: `tidal sync vsphere --config config.yml`

[8] https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-on-a-vps

2) Environment Variables

Another alternative is to set the following environment variables with the needed values and Tidal Tools will use them:

- VSPHERE_USERNAME
- VSPHERE_PASSWORD
- VSPHERE_SERVER
- VSPHERE_TLS
- VSPHERE_INSECURE

After having installed Tidal Tools, begin to sync your inventories with `tidal sync servers`. Tidal sync supports many server inventory management tools such as VMWare, HyperV, and more.

What is Syncing?

Syncing is a process that transfers your inventories to your Tidal Migrations account. When importing your servers to the API, Tidal Migrations's sync tool will check for existing servers, based on their hostname and update the changed data for those servers. If the given server to sync does not exist already, it will add that server to the Tidal Migrations platform.

How do I manually sync my servers?

You can sync any data source with Tidal Migrations by generating a simple JSON document of the data.

This data can be passed as standard input to the `tidal sync servers` command and your servers data will be synchronized via the API.

The JSON document must be created in the specified format. The top-level key must be "servers", with a value of an array. The array can consist of the various keys as shown below, describing the server to be synced. You can also include any other arbitrary fields in the key "custom_fields".

```
{
  "servers": [
    {
        "host_name": "ewrfceapcfg03",
        "description": "This is a general description for this server. The server has several functi
        "custom_fields": {
          "tcp_port": 11441,
          "database_software": "SQL 2008 SP3",
          "database_version": "10.0.5538.0"
         },
        "fqdn": "ewrfceapcfg03.com",
        "environment_id": 2,
        "assigned_id": "198",
        "zone": "Data",
        "ram_allocated_gb": 8,
        "storage_allocated_gb": 83.8,
        "storage_used_gb": 52.06,
        "cluster_id": 48337,
        "role": "Administrator",
        "cpu_count": 4,
        "ram_used_gb": 2,
        "virtual": true,
        "environment": "Production",
        "cluster": {
          "host_name": "rrfedfds"
        }
    }
  ]
}
```

The synchronization of your servers to Tidal Migrations can be performed with the following command:

```
cat some_file.json | tidal sync servers
```

You can easily set this to run periodically so that your servers are synced on a daily basis and the data is up to date. This is a great resource[9] on setting the command as a cron job.

We recommend that you setup your inventories to sync *every 24 hours*, this will keep your resource inventory up to date and accurate over time.

Transforming your data

If your document is not formatted as the above, not to worry.

Suppose you have a **csv** file.

Below is a **sample** Ruby script *transform.rb* that will read the data within the file passed in as standard input, transform it to the JSON format above and output it as standard output.

```
#!/usr/bin/env ruby
require 'json'
require 'csv'

def to_bool(str)
  if str == "true" || str == "yes"
    str = true
  else
    str = false
  end
  str
end

def transform(input)
    # convert input to proper csv format
    csv = CSV.parse(input,{ encoding: "UTF-8", headers: true, header_converters: :symbol, converters
    # convert csv to hashmap with key value pairs
    json = csv.map { |row| row.to_hash }
    data = {servers: []}
    json.each do |vm|
        props = {}
        props[:cluster] = {:host_name => vm[:cluster_host_name]}
        props[:custom_fields] = {:tcp_port          => vm[:custom_fields_tcp_port],
                                 :database_software => vm[:custom_fields_database_software],
                                 :database_version  => vm[:custom_fields_version]}

        props[:assigned_id]  = vm[:assigned_id].to_s
        props[:virtual] = to_bool(vm[:virtual])
        c = vm.merge(props)
        data[:servers].push c
    end
    # display data in pretty json format
    puts JSON.pretty_generate(data)
end
data = STDIN.read
transform data
```

The script above is an example of how to easily transform your data into the necessary JSON object. It can be altered to work with your data or it could be rewritten in any language of your choice.

Change the file permissions to make the script executable using:

```
chmod +x ./transform.rb
```

[9] https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-on-a-vps

You can now utlise your script with a given CSV file and it will be synced to your account via the API. Utlise the command below:

```
cat some_file.csv | ./transform.rb | tidal sync servers
```

How do I sync my servers in more automated fashion?

Instead of manually providing all of your servers details, you can utilize Machine Status[10] utility to gather all the stats from your servers environment. Follow the Machine Stats installation and configuration steps available on the tool's GitHub page[11].

How do I sync other resources?

You also have the option of syncing your **Applications** and **Database Instances**.

Sync your Applications

You can sync your Applications with the following command:

```
cat some_file.json | tidal sync apps
```

When importing your applications to the API, Tidal Migrations's sync tool will check for existing applications, based on their name, and update the changed data for those applications. If the given application to sync does not exist already, it will add that application to the Tidal Migrations API.

The syncronization of your Applications can be performed by following the above procedure with a simple JSON document of the data:

```
{
"apps": [
    {
      "name": "App_name",
      "custom_fields": {
        "Technologies": "Approval Management System DB"
      },
      "description": "This is a general purpose application that has several functionalities. The fi
      "servers": [{
        "host_name": "trpewrcapbiz02"
      }],
      "urls": "https://approvalmanagementsystem.com",
      "transition_overview": "this is the transition_overview of the application",
      "transition_type": 3,
      "transition_plan_complete": true,
      "source_code_location": [
            "/filepath/location",
            "folder1/file1"
      ]
    }
  ]
}
```

Note

This[a] is a similar script as the one above to transform your data into the necessary JSON object for applications.

[a] https://github.com/tidalmigrations/data-transform/blob/master/scripts/csv_transform.rb

[10] https://github.com/tidalmigrations/machine_stats
[11] https://github.com/tidalmigrations/machine_stats

Sync your Database Instances

You can sync your Database Instances with the following command:

```
cat some_file.json | tidal sync dbs
```

When importing your database instances to the API, Tidal Migrations's sync tool will check for existing database instances, based on their name, and update the changed data for those database instances. If the given database instance to sync does not exist already, it will add it to the Tidal Migrations API.

The syncronization of your Database Instances can be performed by following the above procedure with a simple JSON document of the data:

```
{
  "database_instances" : [
  {
    "created_at": "2018-05-25T05:01:48.533Z",
    "updated_at": "2018-05-25T05:01:48.588Z",
    "name": "720 TASK DB",
    "database_size_mb": 1870,
    "database_path": "C:\\system\\databases\\720_TASK_DB",
    "description": "This is a general description for this database instance. This database primaril
    "custom_fields": {
            "Technologies": "Approval Management System DB"
      },
    "environment_id": 2,
    "move_group_id": 3
  }
 ]
}
```

Web applications discovery, analysis and importing

Discover Your Applications

The first step in your cloud migration project is discovering what you have. Thousands of domains are registered on a daily basis and it can be hard to remain informed. Utilize the `tidal discover` tool with your customized Discovery Plan to obtain both private and public domains registered for your given datacentres.

- Scan multiple networks and DNS services with a *discovery plan*

```
tidal discover my_plan.yml > my_urls.txt
```

Note

In order to utilize this command, install DNS Tools[a].

[a] https://dnstools.ninja/download/

With this command, Tidal Discover will output a set of FQDNs for your defined discovery plan and store it in the file *my_urls.txt*.

Your Discovery plan is a YAML[12] file which can include three different ways that you want to scan your networks and DNS services. You may choose to provide a DNS service to extract information, a *named.conf* file for binary configuration, or a collection of zone files to be scanned and generate all the affected domains.

via DNS Service

An example of a discovery plan to obtain FQDNs by specifying a DNS Service.

The file *my_plan.yml* must be of the following format:

```
discovery:
  - name: Q9 Datacenter front-ends
    networks: 10.83.2.0/24
    tcp_ports:
      - 80
      - 443
    dns_service: aws
```

via Bind Configuration

An example of a discovery plan to obtain FQDNs by specifying a named.conf file.

The file *my_plan.yml* must be of the following format:

```
discovery:
  - name: NYC Datacenter front-ends
```

[12] http://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html

```
    networks:
      - 10.83.3.0/24
      - 10.130.241.0/24
    tcp_ports:
      - 80
      - 443
    path_to_bind: "/etc/bind/named.conf"
```

via Zone files

An example of a discovery plan to obtain FQDNs by specifying a zone file.

The file *my_plan.yml* must be of the following format:

```
discovery:
  - name: Tokyo flat-network
    networks: 192.168.0.0/16
    tcp_ports:
      - 80
      - 443
      - 8080
      - 8443
    zonefiles: "~/tokyo_zones/*/**"
```

You may also choose to include all three of the ways in your Discovery plan like so:

```
discovery:
  - name: Tokyo flat-network
    networks: 192.168.0.0/16
    tcp_ports:
      - 80
      - 443
      - 8080
      - 8443
    zonefiles: "~/tokyo_zones/*/**"

  - name: NYC Datacenter front-ends
    networks:
      - 10.83.3.0/24
      - 10.130.241.0/24
    tcp_ports:
      - 80
      - 443
    path_to_bind: "/etc/bind/named.conf"

  - name: NYC Datacenter front-ends
    networks:
      - 10.83.3.0/24
      - 10.130.241.0/24
    tcp_ports:
      - 80
      - 443
    dns_service: aws
```

You can also combine all as the following:

```
discovery:
  - name: NYC Datacenter front-ends
    networks:
      - 10.83.3.0/24
      - 10.130.241.0/24
    tcp_ports:
      - 80
```

```
    - 443
path_to_bind: "/etc/bind/named.conf"
zonefiles: "~/path/to/my/zonefiles"
dns_service: aws
```

Creating your Discovery Plan

Here is some brief information regarding the keys defined in the *my_plan.yaml* file:

Key	Information	Format
`networks`	One or more subnets that you want to include in the process.	Cidr block notation[13]
`name`	A friendly name for your network, e.g. “Tokyo DC-1 Front-End”	Text
`tcp_ports`	One or more TCP Ports that you frequently run web servers on and would like to interrogate: e.g. 80,443,8080,8443 etc.	Integer
`path_to_bind`	The location of a named.conf file[14] for a bind server configuration.	File Path
`dns_service`	Name of a DNS service to be analyzed with DNS tools, currently only “aws” service is avaliable which extracts information from Amazon Route 53 zones.	“aws”
`zonefiles`	The location of a zone file[15] which contains a list of DNS records with mappings between domain names and IP addresses.	File Path

Note

`networks`, `name` and `tcp_ports` are required keys that you must include. Specify one or more of `path_to_bind`, `zonefiles` **or** `dns_service` in your Discovery file.

Be sure to verify the outputted FQDNs that you’d want to analyze.

Next Step

Having discovered your applications, here is a guide on analyzing your FQDNs. Tidal Analyze will review the outputted FQDNs and give you a detailed analysis on what technologies are being in use for each domain.

Tidal Analyze Web

Now that you have discovered your FQDNs for your specified Discovery Plan, the next step is to rapidly capture what technologies are in use, on which networks and with what DNS configuration.

Tidal Analyze Web will fingerprint the technology on both your internet sites and intranet applications behind your firewall in seconds, without needing to install agents. Whether you have one or one million end points, Tidal Tools will centralize the data gathered in our platform for you to analyze.

Simplify your application centric discovery with Tidal Analyze Web.

Gathering your domains and URLs

Simply save a list of URLs or FQDNs in a text file and use that as input.

Or use the discovery command to scan your network and DNS to gather a list of relevant domains.

Analyzing FQDNs and URLs

Now that we have our relevant domains and URLs in *my_urls.txt*, we can analyze them with:

```
tidal analyze web my_urls.txt --upload
```

Tip

You can also utilize the standard output of URLs and FQDNs from Tidal Discover as input. `tidal discover my_plan.yml | tidal analyze web --upload`

Are you running this behind a firewall or in a private network? No problem, drop the `--upload` flag, continue on and then checkout the Uploading to your account section below.

Uploading to Tidal Migrations API

After recieving the results of Tidal Analyze Web, you may or may not be connected to the internet.

If you are so, you can import it to your Tidal Migrations account with the following flag:

```
tidal analyze web my_urls.txt --upload
```

If you aren't connected to the internet, you also have the option to save the results in a JSON file to import at a later time. This can be done with the following steps:

1. Run this command to run the analysis and save the results to the file *analyze_output.json*:

   ```
   tidal analyze web my_urls.txt --type json > analyze_output.json
   ```

2. Copy the file, *analyze_output.json* and install Tidal Tools, on a computer with internet access.
3. Login to Tidal Tools with `tidal login`
4. Run this command to upload your previously generated data to your Tidal Migrations account:

   ```
   tidal analyze web --upload-file analyze_output.json
   ```

Accessing your domains in the Tidal API

Once you have imported the results to the Tidal API, your domains will appear on the right hand side navigation bar under

Assess > **URLs**.

Troubleshooting

It looks like the server did not respond for 10 seconds

If you got this error message you can try to increase the waiting time using the `--timeout` flag and the amount of seconds (default is `10`). For example, `--timeout 30` sets the time waiting for the server to respond to 30 seconds.

Host discovery with Nmap

As part of your cloud migration journey, it is important to have all the tools at your disposal. To facilitate the discovery process, you can use Nmap[16] in addition to the other discovery methods[17].

With Nmap's powerful capabilities and good documentation[18], you can identify all the hosts you have running on your network, open ports, and running services. It is possible to discover those places which are often hard to reach.

Nmap is capable of producing its output in an XML file. It allows you to inspect the raw scan output before sending it to the Tidal Migrations API with Tidal Tools[19]. Once uploaded to the Tidal Migrations Platform, you will be able to visualize your network devices, track your complete server inventory, and build on this data with other discovery methods. This is how you can make informed decisions on your cloud migration path.

Using Nmap with Tidal Tools

By leveraging the power of **Tidal Tools**, you can send the output generated by Nmap to your Tidal Migrations account.

1. Install Tidal Tools[20]
2. Connect Tidal Tools and your Tidal Migrations account with tidal login[21].
3. Run Nmap with the flags of your choosing and save the output to an XML file. For example,

   ```
   sudo nmap -sV -p80,443,8080,8443,1433,1521,27017 <ip-address/range> -oX my-network.xml
   ```

   *Note: the **-sV** flag will attempt to determine the version of the service running on port and the **-oX** specifies the output as an XML file. Want more scanning options?*

   Tip

   Be sure to replace with the CIDR range for the network you would like to scan, ex. `10.0.0.0/24`
4. Run this Tidal Tools command to upload your previously generated Nmap output to your Tidal Migrations account `tidal sync nmap my-network.xml`
5. Head over to your Tidal Migrations account! (https://<>.tidalmg.com/#/discover/host_discovery)

Installing Nmap

Nmap ("Network Mapper") is a free and open source utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

[16] https://nmap.org/
[17] https://guides.tidalmg.com
[18] https://nmap.org/book/host-discovery-find-ips.html
[19] https://get.tidal.sh
[20] https://guides.tidalmg.com/tidal-tools.html
[21] https://guides.tidalmg.com/tidal-tools.html#login

You can find all the documentation[22] and instructions on how to download[23] Nmap to your environment on the official site[24].

Nmap usage

Currently, Tidal Migrations supports the collection of Hosts (IP addresses), PTR records, their open ports, the ports status, the port protocol (TCP/UDP), and the services running in the port, including the version.

Nmap offers a wide range of utilities and commands, such as Port scanning, Host discovery, Service and version detection to name a few. Here are some basic examples for how to do some nmap scanning.

Note

Some commands require super-user privilege

Target Specification

```
nmap 192.168.1.1-254                # Scan a range
```

Scan Techniques

```
nmap 192.168.1.1 -sS                # TCP SYN port scan (Default)

nmap 192.168.1.1 -sT                # TCP connect port scan (Default without root privilege)

nmap 192.168.1.1 -sU                # UDP port scan
```

Host Discovery

```
nmap 192.168.1.1-3 -sL              # No Scan. List targets only

nmap 192.168.1.1/24 -sn             # Disable port scanning. Host discovery only.

nmap 192.168.1.1-5 -Pn              # Disable host discovery. Port scan only.
```

[22] https://nmap.org/docs.html
[23] https://nmap.org/book/install.html
[24] https://nmap.org/

Database analysis

Not sure how ready you are to move to the cloud? With Tidal Migrations you have the option to analyze the databases associated with your applications.

The analysis will calculate the difficulty of migrating your databases to each target platform, and give details on database features that may complicate the migration.

It is capable of analyzing **Oracle**, **SQL Server**, **MySQL**, and **PostgreSQL** databases. Providing analysis on migrating a database to a **variety of services on AWS, Azure and Google Cloud**.

Note

The entire analysis never queries or reads user or application data and does not collect database source code.

Supported Database Versions

Tidal Tools is able to analyze databases with the following versions:

Oracle	SQL Server	MySQL	PostgreSQL
Beta Oracle Database 8i (8.1)	SQL Server 2008R2	5.6	8
Beta Oracle Database 9i Release 2 (9.2)	SQL Server 2016	5.7	9
Oracle Database 10g Release 2 (10.2)	SQL Server 2017	8.0	10
Oracle Database 11g Release 1 (11.1)			11
Oracle Database 11g Release 2 (11.2)			12
Oracle Database 12c Release 1 (12.1)			13
Oracle Database 12c Release 2 (12.2)			
Oracle Database 18c (18.1)			

If you have a use case for a different version, definitely let us know at info@tidalmigrations.com, we are always adding new capabilities.

Migration Complexity

The databases are analyzed to look for patterns and feature usage that may be difficult to migrate due to lack of support or compatibility in their new environment. The databases are analyzed based on their metadata, looking at specific schema objects that are used within your databases as well as the usage of proprietary features that will not be available in the target platforms.

For example, in Oracle databases, the Data Dictionary and AWR repository tables are read and analyzed. The scoring is calculated based on the type of attributes, features or schema objects that are used and the frequency of use throughout the database.

- Over 100 unique characteristics are considered
- Feature-fit is executed against all supported cloud data platforms.
- Migration difficulty score is calculated based on a weighted model

Getting Started

- Enable the Database Analysis feature for your account: https://yoursubdomainhere.tidalmg.com/#/settings - See the Database Analysis section, under Preferences.
- `tidal login` Be sure to have installed and logged in to your Tidal Migrations account via Tidal Tools.
- Install Docker CE[25], it is compatible with most OSs, select the one you need. Version 17.12 or later will work with Tidal Tools. Why Docker?
- You will also need a few authentication and configuration details for the database:
  - `id` - The id of the database from your Tidal Migrations account. You can find it in the URL bar when looking at a database instance. ex. If you are viewing a database instance in Tidal Migrations, the URL will show https://demo2.tidalmg.com/#/database_instances/111 in this case 111 is the database instance ID.
  - `engine` - The database vendor, either `Oracle`, `SQL Server`, `MySQL`, or `PostgreSQL`, it is not case sensitive.
  - `host` - The hostname of the server that the database is located on and is accessible via a network connection from your current device and location.
  - `port` - The port that the host has open and the database can accept connections on, the default for Oracle is `1521`, for SQL Server it is `1433`, for MySQL it is `3306`, and for PostgreSQL the default port is `5432`.
  - `db_name` - The name of the database that will be analyzed, as it is defined within the database engine itself. ie. the value that is used by applications to connect to the database by name.
  - `user` - A username to authenticate with the database with, see below for more details.
  - `password` - A password for the corresponding user.
  - `name` - A common name for your database could be the same or different from db_name, but this value is arbitrary and only for your reference.

Oracle User

You can can create a user with the script and set of permissions defined in this script. You should provide a secure password on the first line.

If you are using a CDB database you will also need to create a second user to access the CDB. You can do that with this script and also provide a secure password at the top:

```
CREATE USER c##tidal_comm_user IDENTIFIED BY "replace_this_with_secure_password" account unlock;
GRANT CREATE SESSION to c##tidal_comm_user;
GRANT SELECT ON gv_$archive_dest to c##tidal_comm_user;
GRANT SELECT ON gv_$instance to c##tidal_comm_user;
GRANT SELECT ON v_$managed_standby to c##tidal_comm_user;
GRANT SELECT ON v_$database to c##tidal_comm_user;
GRANT SELECT ON dba_hist_sysmetric_summary to c##tidal_comm_user;
```

SQL Server User

You can can create a user with the script and set of permissions defined in this script. You should provide a secure password near the top.

MySQL Server User

For MySQL you can create a user (`tidal`) with all the necessary permissions as the following:

```
CREATE USER 'tidal'@'%' IDENTIFIED BY 'replace_this_with_secure_password';
GRANT PROCESS,REFERENCES, SHOW DATABASES, SHOW VIEW ON *.* TO 'tidal'@'%';
GRANT SELECT ON sys.* TO 'tidal'@'%';
GRANT SELECT ON performance_schema.* TO 'tidal'@'%';
GRANT SELECT ON mysql.slave_master_info TO tidal;
GRANT SELECT ON mysql.slave_relay_log_info TO tidal;
GRANT SELECT ON mysql.user TO tidal;
```

Use the commands from the following sections according to the MySQL version used.

[25] https://docs.docker.com/v17.12/install/

MySQL 5.x

```
GRANT SELECT ON mysql.proc TO tidal;
```

MySQL 8.x

```
GRANT SHOW_ROUTINE ON *.* TO 'tidal'@'%';
GRANT SELECT ON mysql.user TO 'tidal'@'%';
```

PostgreSQL Server User

Use the following commands to create a user (`tidal`) on PostgreSQL server.

```
CREATE USER tidal WITH PASSWORD 'replace_this_with_secure_password';

DO $$ DECLARE comm_rec RECORD;
BEGIN
    FOR comm_rec IN
                SELECT 'GRANT REFERENCES ON ALL TABLES IN SCHEMA '||schema_name||' TO tidal' AS comm
        LOOP
        EXECUTE comm_rec.comm;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

GRANT REFERENCES ON ALL TABLES IN SCHEMA public to tidal;
```

For PostgreSQL versions **higher than 9** the following `GRANT`s should be also applied:

```
GRANT SELECT ON pg_catalog.pg_config TO tidal;
GRANT EXECUTE ON function pg_catalog.pg_config TO tidal;
GRANT SELECT ON pg_catalog.pg_proc TO tidal;
GRANT SELECT ON pg_catalog.pg_namespace TO tidal;
```

After creating the user you will need to add the appropriate entry to the `pg_hba.conf`[26]. For example:

```
# TYPE  DATABASE    USER    ADDRESS     METHOD
host    all         tidal   0.0.0.0/0   md5
```

To apply the configuration changes to the running PostgreSQL server you will need to run `pg_ctl restart`[27].

Perform the analysis

With your user and password, you can define all these values in a YAML configuration file.

The simplest way is to use `tidal analyze db init` and answer the questions. Or you can create the file manually:

databases.yaml:

```
databases:
  - id: 111
    engine: Oracle
    host: 'my-db-host.com'
    port: 1521
    db_name: 'orcl'
    user: 'tidal'
    password: 'yoursecurepassword1234!'
    name: 'My-Test-DB'
```

[26] https://www.postgresql.org/docs/current/auth-pg-hba-conf.html
[27] https://www.postgresql.org/docs/current/app-pg-ctl.html

Note

It is best to use quotations, either double or single, around the values in the configuration file. To avoid special characters, : `{ } [ ] , & * # ? | - < > = ! % @ \ \n` from being interpreted. Single quotes are safest, if the value has a single quote within it, you can include it by using a two single quotations, ie. `'my''string'` - will become `my'string`.

Tip

Are you analyzing an Oracle Stanard Edition (SE) database or using a CDB database? Check out the advanced configuration below.

- You're all set! You can now analyze the database with:

```
tidal analyze db databases.yaml
```

Try it out!

Running offline

If you need to run the command from a computer without any internet access, either no download access to download the docker image necessary or no outbound access to upload the results of the analysis to the API then this is for you.

First you will need to setup Tidal Tools on a machine with internet access. Next you can run:

```
tidal backup
```

This will create a tar file called `tidal-snapshot_DATE.tar`

Moving to the air-gapped machine you will need to install Tidal Tools and Docker and transfer the tar file above, then run:

```
tidal restore tidal-snapshot_DATE.tar
```

This will load the docker image and all of existing Tidal Tools configurations from the original machine. You can now run the database analysis without any external network connectivity, except to your database host itself:

```
tidal analyze db --offline databases.yaml
```

This will output a zip file called, `tidal-dba-results_DATE.zip` that can then be uploaded to the application for a given database in order to complete the analysis:

```
tidal analyze db --upload tidal-dba-results_DATE.zip
```

You should recieve confirmation that the upload has completed and can navigate to Tidal Migrations to see the results.

Why Docker?

You need to install Docker in order to complete the database analysis. This is because the analysis uses several system dependent software libraries, so by using Docker the analysis can use those libraries without you requiring to install the correct dependencies with the correct versions.

What about security?

The entire analysis takes place locally on your machine. The only data that is captured and sent from the analysis are the results of the analysis and metadata. No application data, source code, files or the contents of any files on your machine are ever copied or sent anywhere.

Advanced Configuration

Oracle Standard Edition

To use Oracle features included only in the Oracle Standard Edition (SE) license, you can set the `analyze_workload` property to `false` in your configuration file. For example:

```
databases:
  - id: 111
    analyze_workload: false
    engine: Oracle
    host: 'my-db-host.com'
    port: 1521
    db_name: 'orcl'
    user: 'tidal'
    password: 'yoursecurepassword1234!'
    name: 'My-Test-DB'
```

Note

By setting this to `false`, some results from the analysis will not be available, including server metrics, connected applications, and the workload analysis ranking.

Troubleshooting

If you are getting errors when trying to perform the analysis, it can help if you confirm that you do have network connectivity to the database.

Two commands you can use for this are:

1. `dig your_db_host` - This should return a DNS record, usually an A record, with an IP address. This means you are able to resolve the hostname of the database. If there is no IP address then you either need to adjust the hostname or you need to configure or adjust the DNS server for your operating system.
2. `nc -vzn -w 10 your_db_host db_port` This command should return `Connected to your_db_host:db_port` if it is able to connect. If it returns `Connection Timed Out` this means that it is not able to reach your database on this port. This could mean that you do not have the correct network connectivity to the database and may need to adjust firewalls or other network access. Or you are you not providing the correct port that is open and listening for requests on the database.

CDB Configuration

You will need these additional values added to your configuration file to connect to a CDB database.

```
databases:
  - id: 111
    engine: Oracle
    host: 'my-db-host.com'
    port: 1521
    db_name: 'orcl'
    user: 'tidal'
    password: 'yoursecurepassword1234!'
    name: 'My-Test-DB'
    dbidtype: 'dbid'
    ispdb: false
    cdb:
      name: 'cdbname'
      user: 'c##tidal_comm_user'
      password: 'your_secure_password'
```

Source code analysis

Tidal Analyze Source Code

Not sure how ready you are to move to the cloud? With Tidal Migrations you have the option to analyze your specified source code associated with the applications.

The analysis will identify the difficulty to migrate your applications to the cloud, including the number of blocking issues identified. It is able to analyze source code written in C#, COBOL, Java, JavaScript, Kotlin, Microsoft Transact-SQL, PHP, Python, Swift, TypeScript, VB.Net.

If you are interested in a deeper dive source code assessment, let us know at info@tidalmigrations.com.

Analytics

Roadblocks

Roadblocks slow down your cloud migration journey, by identifying the total number of roadblocks in your source code, you are able to instantly determine which applications should be prioritized over others from a technical difficulty perspective.

Migration Difficulty

Migration difficulty tracks each application's readiness for a Paas system in the cloud. This percentage lets you identify which applications are easier or harder based on the implementation details and application structure.

With these 2 key numbers, stay alert and build the smartest roadmap to the cloud.

Criteria

To determine how ready an application is for a cloud migration there are **over 190 different factors** across all the languages (C#, VB/VB.net, Java, T-SQL, Python and PHP) that are considered. An example of some of the criteria that are considered include: - Using the file system - Using system DLLs - Using hardcoded IP addresses - Using Access Control Lists

The analysis will look at all of these factors and determine an overall migration difficulty.

Getting Started

After having installed Tidal Tools, analyze your source code and rank your applications by *Migration Difficulty*.

- Enable the Source Code Analysis feature for your account - https://yoursubdomainhere.tidalmg.com/#/admin/settings - at the bottom.
- Be sure to have logged in to your Tidal Migrations account via Tidal Tools.
- Install Docker CE[28], it is compatible with most OSs, select the one you need. Version 17.12 or later will work with Tidal Tools. Why Docker?

[28] https://docs.docker.com/v17.12/install/

- You will need the application ID - You can find it in the URL bar when looking at an application. ex. If you are viewing an application in Tidal Migrations, the URL will show https://demo2.tidalmg.com/#/apps/111 in this case 111 is the application ID.
- You will also need a local copy of the source code for the application.

Tip

Looking to try it out and don't have any code handy? You can use this sample schoolbus application[a] by cloning it from GitHub[b].

[a] https://github.com/tidalmigrations/schoolbus
[b] https://help.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository

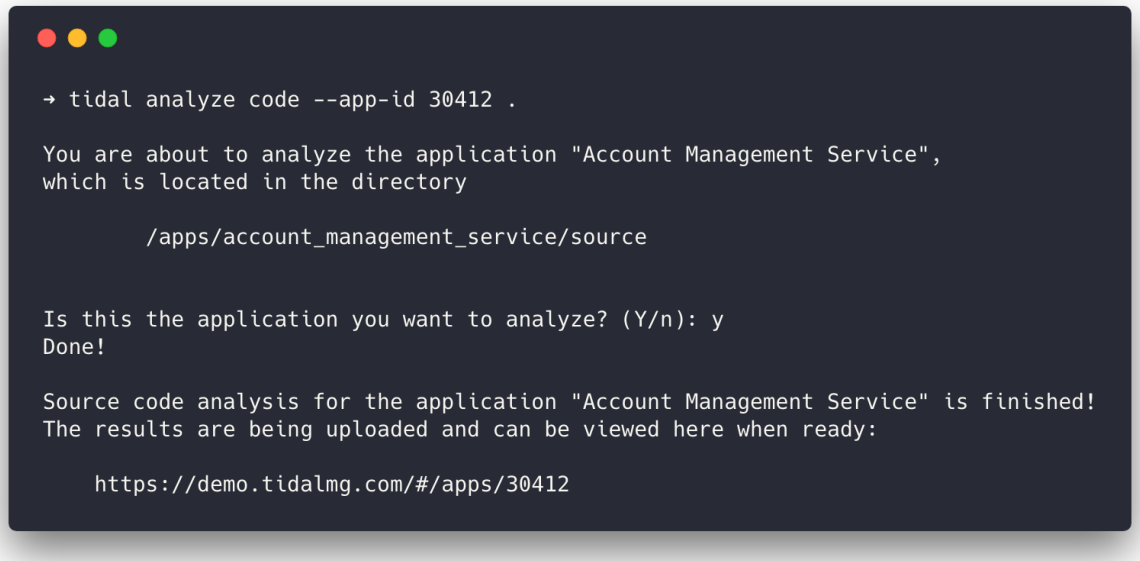
You are all set, you can now analyze the source code with:

```
cd /path/to/source-code
tidal analyze code --app-id 111
```

You can get the ID for *your* application from the end of the URL, in the address bar in your browser, when viewing the application in Tidal Migrations online.

Once it is complete you can view your application and the newly updated data in Tidal Migrations, it can take a couple minutes for the data to be processed and uploaded.

Try it out!



Analyze your source code

Why Docker?

You need to install Docker in order to complete the source code analysis. This is because the analysis uses several system dependent software libraries so by using Docker the analysis can use those libraries without you requiring to install the correct versions and dependencies.

What about security?

The entire analysis takes place *locally on your machine*. The **only** data that is captured and sent from the analysis are the results of the analysis and metadata. **No source code, files or the contents of any files on your machine are ever copied or sent anywhere.**

Summary

Congratulations! You have completed your discovery across your portfolio.

We hope that this book helped you to get yourself familiar with **Layering Discovery Techniques**:

1. Importing spreadsheets of applications, databases, servers, and virtualization clusters
2. Scheduling synchronizations across your hypervisors such as VMWare or Hyper-V
3. Server usage statistics aggregation
4. Discovering and fingerprinting your running web applications
5. Performing databases analysis
6. Analyzing your source code

That was the beginning of your journey towards the transformative cloud migrations. And now, we hope, you are ready to move forward!

Next Steps

Now that you have completed your discovery across your portfolio, you are ready to do your assessment. The first step is to make sure you have a set of discrete applications in scope for your migration project. Navigating to your list of `Applications` under the `ASSESS` section in the platform, you should be able to see a list of all your applications. From here, you can tag them with a specific tag for your project, and even create a dashboard tile that includes all the items in scope.

With the list of applications and a full set of data from the discovery process, you can now perform an initial assessment and determine a transition strategy (one of the 6 Rs) for each application.

As well with this data, you can generate a report, from the `Portfolio Reports` section under `PLAN`, in order to get a full report on your assessment. This report can be shared with other parties and people interested in the project progress.

Appendix

Technical troubleshooting

Most problems with Tidal Tools can be fixed by following the troubleshooting methods described below. If you need extra help with any of this, you can reach us at support@tidalmigrations.com.

General troubleshooting options

Diagnose dependencies and environment with `tidal doctor`

To work properly and to provide some of its features Tidal Tools depends on some additional software and external services. To check your environment run `tidal doctor` and review its output for any warnings and ways to recover.

Sample `tidal doctor` output looks like the following:

```
[-] Tidal Tools v2.2.20
    X Updates available.
      Go to https://get.tidal.sh/ for update instructions.
    • Config file in use: /home/tidaluser/.config/tidal/config.yaml
    • Logfile location: /home/tidaluser/.local/share/tidal/tidal.log

[√] Tidal API connection
    • Tidal API connection configured OK.

[√] Docker
    • Docker at /usr/bin/docker
    • Server 18.09.0 • API 1.39 (min. 1.12) • Client 1.39
    • Pull from registry OK

[-] DNS Tools
    X DNS Tools not installed.
      Go to https://dnstools.ninja/download/ for installation instructions

[-] vSphere connection
    X vSphere connection not configured.
      Run 'tidal login vsphere' to set it up.
```

Update Tidal Tools

Periodically Tidal Tools checks if the newer version is available. It would inform you about it as a message printed to your terminal or command prompt output after any `tidal` command invocation:

```
Looks like you are running an older version of Tidal Tools.
Check https://get.tidal.sh for update instructions!
```

Also you can explicitly check for new Tidal Tools versions available by running `tidal check-updates` or `tidal doctor` command.

Default log file location

To perform better diagnosis of any issues with Tidal Tools, some of the underlying activities performed by Tidal Tools are written to the log file. If you are having trouble with one of the Tidal Tools commands, you can reach us at support@tidalmigrations.com and send us your log file for us to investigate.

The default locations of the log file are:

- `C:\Users\tidaluser\AppData\Roaming\tidal\tidal.log` on *Windows*
- `/home/tidaluser/.local/share/tidal/tidal.log` on *Linux*
- `/Users/tidaluser/Library/tidal/tidal.log` on *macOS*

Please note that log files are truncated by default before the `tidal` invocation. That means that the default log file contains entries specific to the one particular `tidal` run.

How to prevent log file from truncating

The default log files truncation behavior may be not desirable in cases when it is necessary to combine log entries of the several subsequent `tidal` invocations. To prevent the existing log file from truncating you can utilize `--keep-log` command line flag. When the `--keep-log` flag is used the log file (default, or one specified with `--log-file`) won't be truncated before the `tidal` command call.

Docker installation

Some of the Tidal Tools features (for example, `tidal analyze code`) depend on Docker to be installed. To get the Docker installation instructions please check the Docker Documentation[29].

How to check if Docker actually works

There are a few ways to check if your Docker installation actually works.

Open your terminal emulator or command prompt and run the command `docker run hello-world`. The sample output should look like the following:

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
…
```

To explicitly check that your Docker installation is able to communicate with Tidal Migrations container registry run the command `docker run gcr.io/tidal-1529434400027/hello-world`. You should see the output similar to the following:

```
Unable to find image 'gcr.io/tidal-1529434400027/hello-world:latest' locally
latest: Pulling from tidal-1529434400027/hello-world
9e91b00c0251: Pull complete
Digest: sha256:b60c2de90e0b5c4f7e74b84ca888e2fec1d288d47c99e48bd612e0eefeb604c5
Status: Downloaded newer image for gcr.io/tidal-1529434400027/hello-world:latest
Hello from Tidal Migrations!
```

Another option is to run `tidal doctor` and check the Docker section of the output.

Docker networking issues

Sometimes Docker has issues with DNS[30] and you may see an error like:

```
Error response from daemon: Get https://gcr.io/v2/: proxyconnect tcp: dial tcp: lookup http on 192.1
```

[29] https://docs.docker.com/install/
[30] https://docs.docker.com/docker-for-windows/troubleshoot/#networking-issues

If so you can update the DNS server used by docker[31] to `8.8.8.8`, as Docker recommends, to solve the issue.

Specifying Docker Proxy

If you need a proxy to access the internet and see an error that looks similar this:

```
Error response from daemon: Get https://gcr.io/v2/: net/http: request canceled while waiting for con
```

You may need to configure Docker to use the proxy server[32].

NB: If you need to authenticate with the proxy, be sure to include the username and password in the value, ie. 'http://proxy_userid:proxy_password@proxy_ip:proxy_port'

Windows specific troubleshooting

Setting up Docker for Windows to use Linux containers

Docker for Windows supports both Linux containers and Windows containers, however Tidal Tools works when your Docker installation is set to to use Linux containers.

To check if your Docker installation was configured to use Linux containers please check one of the following:

- Open Docker for Windows menu and check for the item **Switch to Windows containers**. If you can find it, that means that your Docker is configured to talk with Linux daemon, so no further actions are needed to be performed.
- However, if you see **Switch to Linux containers** in the Docker for Windows menu, that means that the Docker was configured to talk to Windows daemon. You need to click that menu item to switch to Linux containers.
- Running `tidal doctor` also checks if your Docker installation is configured to use Linux containers or not.

Issues running Tidal Tools with PowerShell ISE

Since PowerShell ISE only runs console apps that don't require user input some of the Tidal Tools commands may work incorrect. It's recommended to use PowerShell instead (i.e. `PowerShell.exe`, **not** `PowerShell_ise.exe`).

However if you really need to use PowerShell ISE you should make some preparations. Most of the time Tidal Tools ask user for some input is when it prompts for connection credentials. To prevent Tidal Tools from prompting you should explicitly specify all the necessary connection information. It could be done by either using `tidal config` command, or manually editing the configuration file, or by setting up the appropriate environment variables.

Windows directory separators in YAML files

Windows traditionally uses the backslash (`\`) to separate directories in file paths. For example, `C:\Program Files\Tidal Software\tidal`. However, the configuration and discovery plan language (YAML[33]) also uses the backslash (`\`) as an escape character in quoted strings[34]. This can make it awkward to write literal backslashes.

Generally it is OK to use forward slashes, because most of the time the Windows file system APIs will accept **both** the backslash (`\`) and forward-slash (`/`) in file paths. But when you use backslashes, you must pay extra attention to keep them from being suppressed by YAML string quoting. That means that you must escape the backslash character with another backslash (which is the escape character), so the string must be as the following: `C:\\Program Files\\Tidal Software\\tidal`.

[31] https://docs.docker.com/docker-for-windows/troubleshoot/#networking-issues
[32] https://docs.docker.com/docker-for-windows/#proxies
[33] https://yaml.org/
[34] https://yaml.org/spec/1.2/spec.html#id2776092

Linux troubleshooting

Manage Docker as a non-root user

The Docker daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user `root` and other users can only access it using `sudo`. The Docker daemon always runs as the `root` user.

If you don't want to preface the `tidal analyze code` command with `sudo`, create a Unix group called `docker` and add users to it. When the Docker daemon starts, it creates a Unix socket accessible by members of the `docker` group.

To create the `docker` group and add your user:

1. Create the `docker` group.

```
$ sudo groupadd docker
```

2. Add your user to the `docker` group.

```
$ sudo usermod -aG docker $USER
```

3. Log out and log back in so that your group membership is re-evaluated. If testing on a virtual machine, it may be necessary to restart the virtual machine for changes to take effect. On a desktop Linux environment such as X Windows, log out of your session completely and then log back in.

4. Verify that you can run `docker` commands without `sudo`.

```
$ docker run hello-world
```

"docker: Error response from daemon: OCI runtime create failed" error message on Fedora 31

Fedora 31 is the first major Linux dustribution that comes with cgroup v2 enabled by default. However, Docker still do not support cgroup v2.

To start Docker on Fedora 31 run the following command and reboot:

```
$ sudo dnf install -y grubby && \
  sudo grubby \
  --update-kernel=ALL \
  --args="systemd.unified_cgroup_hierarchy=0"
```

This command reverts the systemd configuration to use cgroup v1.