# BIT Platform

This documentation is for internal project use only and provides important details about the BIT Platform System from Host Details, Common Tasks and configuration.

## Terms:

The following terms will be used throughout this documentation are listed here for convince

* **AWS** – Amazon Web Services, the company in which we use to host our production services
* **DNS** – Domain Name Server
* **Route53** – AWS’ DNS Management for op-bit.nz domain
* **EC2**- AWS’ Virtual Compute Service, Provides Virtual Hosts for us to run services on
* **Docker** – A Virtualization Platform that allows easy deployment of services in a easily configurable way
* **VMWare ESXi** – A Hypervisor host to allow for running of virtual machines
* **VMWare vCenter/VCSA** – A Management tool for Multiple ESXi hosts
* **SimpleSAML** – PHP framework to allow for SAML integration/authentication within apps
* **IDP** – Identity Provider, Provides identity information for users
* **AD** – Active Directory Server, Centralized User Management
* **ADFS** – Active Directory Federation Services, a IDP that reads from active directory
* **SSO** – Single Sign On, 1 account authentications multiple applications
* **Staging Environment** – The internal Development Environment of the BIT Platform
* **OPNET** – Otago Polytechnic Network
* **LetsEncrypt** – Open Source Free HTTPS certificate issuing service

## Important Notes

This Document Contains No Passwords/Private Key files and nor should it, see the Project Leader or EAD SysAdmin should you require them.

As the platform has 2 environments - production and staging there are many IP addresses for staging machines, an internal DNS server has been setup to provide ease of access for group members.

NOTE: The Staging environment is only accessible internal from within OPNET

There are 2 DNS servers to provide redundancy should one fail, the DNS servers are configured to provide routes for \*.staging.op-bit.nz addresses, with fall back to the original OPNET DNS servers to allow machines to still resolve internal \*.op.ac.nz addresses

Primary DNS: 10.118.27.130

Secondary DNS: 10.118.27.132

To be able to resolve the staging.op-bit.nz addresses there has been multiple Client Machines setup in the cluster that you can remote desktop to, there have relevant tools installed along with bookmarks to import staging services.

There have been assigned static IP addresses and are location in the ranges between 10.118.27.120-124.

To login you must specify your OP-BIT/ username and password (this is the one that has been setup in our own domain controller).

To ease account setup within the internal platform apps/services we are running our own internal AD server that allows us to login to all of our internal/staging services with a single login.

Production Services/Apps use our managed IDP which authenticates against OP’s ADFS service

## Service Addresses:

### Staging:

**AD1:** ad1.staging.op-bit.nz / 10.118.27.130 (static)

**AD2:** ad2.staging.op-bit.nz / 10.118.27.132 (static)

**ADFS:** adfs.staging.op-bit.nz / 10.118.27.134 (static)

**GitLab:** gitlab.staging.op-bit.nz

**Gateway:** gateway.staging.op-bit.nz

**IDP:** idp.staging.op-bit.nz

**Mattermost:** mattermost.staging.op-bit.nz

**Ticketing:** ticketing.staging.op-bit.nz

**Monitoring:** monitoring.staging.op-bit.nz

**vCenter:** vcsa.staging.op-bit.nz

### Production:

<https://gitlab.op-bit.nz>

<https://mattermost.op-bit.nz>

<https://idp.op-bit.nz>

<https://dev-iis.op-bit.nz>

<https://355631893594.signin.aws.amazon.com/console>

<https://gitlab.com/op-bit-platform>

### Email:

These are the emails used for the project, and are used for sending/receiving emails related to the project

[opbitplatformadmin@gmail.com](mailto:opbitplatformadmin@gmail.com) – Used for LetsEncrypt, Security Updates of Gitlab & Mattermost, all emails sent to this address are captured by a bot that spits the email out in a Mattermost channel

[bitplatformadmin@op.ac.nz](mailto:bitplatformadmin@op.ac.nz) – Used for sending emails from Gitlab & Mattermost (i.e. confirming email addresses, forgot passwords etc.)

[bitplatformsupport@op.ac.nz](mailto:bitplatformsupport@op.ac.nz) – Used for the ticketing system, all emails sent to this email are captured by the ticketing system

## Server Logins:

Private keys are used for authentication of the Linux hosts

#### Production Machines

Hosted on AWS use the bit-platform-dev.ppk file, with the username **ubuntu**

NB: AWS images do not require a password for sudo

#### Staging machines

Hosted onsite use the bit-platform-staging.ppk file, with the username **user**

NB: this private key is preset in the template, and is required to login, no password is required for sudo

Windows Hosts use username/password (Each user should have their own account on windows) \*should the need arise there is a BITADMIN account available to access the machine.

## New Team Member

When a new Team Member joins the project, they will need to be issued with a few accounts to allow them to login to the various services required for development of the Platform.

### AWS:

The Project Leader/EAD Sysadmin have AWS admin rights and will create new members accounts.

1. Login to AWS Console
2. Click Services -> IAM
3. Select Users on the left Menu
4. Click the Blue Add User Button
5. Enter the username of the user (You can do multiple users clicking the add another user button)
6. Tick AWS Management Console Access for the access type
7. Leave Autogenerated password selected
8. Ensure User must create a new password at next sign-in
9. Click Next Permissions
10. Tick the BIT\_Platform group
11. Click Next Review
12. Click Create User
13. Download .csv and send passwords out to each user

### GitLab.com:

Even though we are hosting our own GitLab we have created a group on GitLab.com to store all important files that in the event of an outage we are still able to access these files for debugging/resolving the issue.

This group is private and requires an invitation to join, the Project Leader or EAD Sysadmin will invite you to join the group.

1. Login to gitlab.com
2. In the top left menu, select Groups
3. Click on op-bit-platform
4. Select Members from the tabs on the top menu
5. Enter the users email address (op email), set the permission as master and click add to group

### Staging Environment:

As all development apps are to use the internal ADFS/LDAP for authentication accounts need to be created for members on 1 of the 2 AD machines hosting in the staging environment.

#### Normal User

1. Login to an AD server (dc1.staging.op-bit.nz/dc2.staging.op-bit.nz)
2. Open Server Manager
3. From the Tools Menu, select Active Directory Users & Computers
4. Expand out the tree and select the BIT-Platform Organizational Unit (OU), and select the Users OU
5. Right Click on Users and Select New -> User
6. Fill in the Details of the user, Providing First and Last names, and use their OP username as the username
7. Select Next
8. Give a Password for the user (Ether get the person to enter the password or randomly generate one for them)
9. Click Next, Then Finish
10. Once the user is created Right Click on them, Select Properties
11. Enter their OP Email address under email (OP Staff is FirstName.LastName@op.ac.nz Students are opusername@students.op.ac.nz)
12. Select the Member of tab, then click the Add Button, Depending on what privileges they need will depend on what groups they are added to, below is a table with groups and the permissions they grant.
13. Enter the names of the groups in the box, click check names then click OK
14. Click OK to save the changes to the User

##### User Groups in the Internal AD

|  |  |
| --- | --- |
| Group Name | Privileges/Description |
| IcingaWebAdmin | Grants access to the Icinga Web interface |
| vCenter-Admin | Grants Administrator access in vSphere Web Client |
| vCenter-User | Grants Read Only Access in vSphere Web Client |
| ZammadAdmin | Grants Admin access in Zammad (Ticketing System) |
| ZammadAgent | Grants Agent access in Zammad (Ticketing System) (Ability to respond to tickets) |

Admin User

Admins are created exactly the same as above except that the user goes in the Administrator OU, Only the full name and username fields need to be filled out, and both fields should follow the naming scheme a-[FirstName] (Should another user with the same first name be in the group then the first letter of their last name should be appended to the end)

They need to be added to the following groups

Domain Admins

Administrators

## DNS Management:

### Staging:

AD1/AD2 can be used to configure DNS

1. Login to ether AD1/AD2
2. Open Server Manager
3. Click Tools
4. DNS Manager
5. Expand the tree
6. BITPDC0X (X being whatever AD you chose)
7. Forward Lookup Zones
8. Staging.op-bit.nz
9. To add a new subdomain right click the middle section, select new Host (A or AAAA)
10. Give a name e.g. gitlab (the subdomain you wish to create)
11. Set the IP address to be the IP address of the host you wish to map that subdomain to
12. Done

To update records just double click on the subdomain you with to update the ip address for

### Production:

The op-bit.nz domain is managed via AWS’s Route 53

1. Login to the AWS Management console
2. Click Route 53
3. Select Hosted Zones
4. Click op-bit.nz.
5. To create a new subdomain click create record set
6. Input the name you wish to be the subdomain e.g. gitlab
7. Ensure type is A –Ipv4 Address
8. In the value field input the ip address of the host you wish to map
9. Leave the rest as default
10. Click create

To update records click the subdomain in the list and update the value ip address to the new address of the host and click save

NOTE: AWS DNS does take a while to apply within OP as OP caches DNS records

## Virtual Machine Creation/Management

### Staging:

vCenter is used to manage our staging cloud environment, this allows you to view, manage and interact with VM’s in our cluster.

Templates are provided for

* Ubuntu 16.04 – This contains the staging key already configured, vCenter will provide a customization wizard to configure the host name when cloning
* Windows Server 2012R2 – This has been template with updates installed and sysprep has been run.

Creating a machine from a template

1. Login to vCenter
2. Select VMs and Templates in the navigation bar
3. Right click on the template you wish to deploy
4. New VM from this template
5. Give it a name
6. Pick a Host to store it on
7. Select the Datastore to store the vm
8. Tick customize this operating system (If it’s the Ubuntu template)
9. Select Ubuntu settings
10. Enter the hostname
11. Done

### Production:

To get machines created in AWS you will need to seek approval from the Project Leader or EAD Sysadmin. This will need to be approved before it will be created.

1. Ensure you are in the correct region (e.g. Sydney)
2. Open EC2 from the AWS management console
3. Click the Launch Instance Button
4. Choose the image (Ubuntu Server 16.04 LTS is what everything uses)
5. Pick a tear (CPU,Memory)
6. Leave the Instance Details as default (unless variation is needed)
7. Add additional Drives or increase the root drive size
8. Add a Name Tag and others if needed
9. Pick an existing security group or configure one with custom port openings
10. Choose an existing Key Pair, bit-platform-dev
11. Launch

## Middle IDP/SSO Service:

IDP is used as a middleman to authenticate users with OP’s ADFS. All our services (GitLab, MatterMost, and Gateway) authenticate against our IDP and our IDP authenticates with OP’s ADFS. Once our IDP authenticates your credentials, it is stored in the IDP session, so from then on you are just authenticated against IDP only (IDP does not authenticate against OP every time).

### Staging:

In staging environment, you can do testing and development of Authentication at <https://idp.staging.op-bit.nz/simplesaml> (Note: This authenticates with our internal ADFS only, not with OP’s ADFS).

### Production:

In production, the IDP is located at <https://idp.op-bit.nz/simplesaml> .

To authenticate the user has to be a member of either ‘grp\_prg\_ICTIN’ or ‘grp\_Staff\_ICT’ group.

There is also an authorization of ‘BITPlatformAdmin@op.ac.nz’ UPN

### Configuration/Management:

MiddleIDP runs simplesamlphp (simplesamlphp.org), it is dockerized for ease of management, there are 3 volumes passed into the container.

The docker files can initial configuration are located here: <https://gitlab.com/op-bit-platform/idp-docker>

#### Adding Services/Apps:

1. Obtain the XML metadata from the service you wish to authenticate (Ex. Gitlab, Mattermost).
2. Go to <https://idp.staging.op-bit.nz/simplesaml> Under “Federation” tab, click on “XML to SIMPLESAMLphp metadata converter” and paste the metadata and click on “Parse”.
3. Copy “saml20-sp-remote” data and go to the IDP machine and navigate to “idp-docker/simplesamlphp/metadata/saml20-sp-remte.php” and paste it at the bottom of the file and save it.
4. Test if it works.

#### Debugging Tips:

User who should be able to login is unable to login and is receiving unauthorized

1. Go to <https://idp.staging.op-bit.nz/simplesaml> and under Authentication click on “Test configured authentication sources”.
2. Click on “staging-sp” / “dev-sp” (production), login with the credentials and see what it throws back, you should be able to figure out where the problem is occurring. (Group membership)

## Ticketing:

TODO (JOSH INSTALLING/SETUP)

## GitLab:

Gitlab receives updates at least once a month, you should receive notification emails should there be any massive security/feature releases.

### Updating Gitlab:

#### Staging:

Staging GitLab updates via the “Pipeline”, simply obtain the new tag of the version number from: <https://hub.docker.com/r/gitlab/gitlab-ce/>

1. Login to gitlab.com
2. Open op-bit-platform/gitlab-docker
3. Change the branch to staging
4. Click the docker-compose.yml file, select edit
5. Paste the new version tag on the 2nd line and replace the existing one
   1. E.g. image: 'gitlab/gitlab-ce:[NEWTAG] ' -> image: 'gitlab/gitlab-ce:9.2.2-ce.0'
6. Put a good commit message & commit the file
7. The pipeline should automatically begin and start downloading the new image to the staging GitLab Machine
8. Click on Pipelines
   1. You should see that your latest commit message and it should be running or waiting
   2. Once stage 1 is complete you can click the play button and select deploy

\*\*Note this will cause staging GitLab to reboot, beware there will be an outage for at least 2 min while it migrates

#### Production:

Production GitLab is a bit more hands on, to update, same as staging obtain the new tag of the version number from: <https://hub.docker.com/r/gitlab/gitlab-ce/>

<https://hub.docker.com/r/gitlab/gitlab-ce/>

1. Login to gitlab.com
2. Open op-bit-platform/gitlab-docker
3. Ensure you are on the master branch
4. Click the docker-compose.yml file, select edit
5. Paste the new version tag on the 2nd line and replace the existing one
   1. E.g. image: 'gitlab/gitlab-ce:[NEWTAG] ' -> image: 'gitlab/gitlab-ce:9.2.2-ce.0'
6. Put a good commit message & commit the file
7. Login to the Gitlab production machine
   1. Cd to gitlab-docker
   2. Run git pull
   3. Login with your gitlab.com credientals
   4. Run docker-compose pull -> this will download the new image
   5. Once that completes run docker-compose up –d -> this will recreate the container and run migrations to the new version

\*\*NOTE: THIS WILL TAKE GITLAB DOWN FOR ATLEAST 2 MINIUTES, ONLY RUN WHEN YOU CAN AFFORD TO HAVE A SHORT OUTAGE.

### Paths & Docker Volumes:

*These paths apply to both staging & production unless specified*

Gitlab is located at: ~/gitlab-docker

Gitlab data is located at /srv/gitlab (Docker volumes are used to maintain data when containers are destroyed and recreated)

* + /srv/gitlab/config -> maps to /etc/gitlab
  + /srv/gitlab/data -> maps to /var/opt/gitlab
  + /srv/gitlab/logs -> maps to /var/log/gitlab

Production Only -> There is also a volume for https certificates (See Http Certificates)

## Mattermost:

Mattermost receives updates at least once a month (16th ish), you should receive notification emails should there be any massive security/feature releases.

### Updating Gitlab:

#### Staging:

Staging Mattermost updates via the “Pipeline”, simply obtain the url to the new tar from: <https://about.mattermost.com/download/> (Ensure its the Enterprise Edition)

1. Login to gitlab.com
2. Open op-bit-platform/mattermost-docker
3. Change the branch to staging
4. Open the app folder
5. Click the Dockerfile file, select edit
6. Paste the new tar tag on the 6th line and replace the existing one
   1. E.g. RUN curl https://releases.mattermost.com/X.X.X/mattermost-X.X.X-linux-amd64.tar.gz | tar -xvz ->
   2. RUN curl https://releases.mattermost.com/3.10.0/mattermost-3.10.0-linux-amd64.tar.gz | tar -xvz
7. Put a good commit message & commit the file
8. The pipeline should automatically begin and start downloading the new image to the staging Mattermost Machine
9. Click on Pipelines
   1. You should see that your latest commit message and it should be running or waiting
   2. Once stage 1 is complete you can click the play button and select deploy

\*\*Note this will cause staging Mattermost to reboot, beware there will be an outage for at least 20 sec while it reloads

#### Production:

Production GitLab is a bit more hands on, to update, same as staging obtain the url to the new tar from: <https://about.mattermost.com/download/> (Ensure its the Enterprise Edition)

1. Login to gitlab.com
2. Open op-bit-platform/mattermost-docker
3. Ensure you are on the master branch
4. Open the app folder
5. Click the Dockerfile file, select edit
6. Paste the new tar link on the 6th line and replace the existing one
   1. E.g. RUN curl https://releases.mattermost.com/X.X.X/mattermost-X.X.X-linux-amd64.tar.gz | tar -xvz ->
   2. RUN curl https://releases.mattermost.com/3.10.0/mattermost-3.10.0-linux-amd64.tar.gz | tar -xvz
7. Put a good commit message & commit the file
8. Login to the Mattermost production machine
   1. Cd to mattermost-docker
   2. Run git pull
   3. Login with your gitlab.com credientals
   4. Run docker-compose build -> this will build the new image
   5. Once that completes run docker-compose up –d -> this will recreate the container and update to the new version

### Paths & Docker Volumes:

*These paths apply to both staging & production unless specified*

Gitlab is located at: ~/mattermost-docker

Gitlab data is located at : ~/mattermost-docker/volumes (Docker volumes are used to maintain data when containers are destroyed and recreated)

* + ~/mattermost-docker/volumes/app/mattermost/config -> maps to /mattermost/config
  + ~/mattermost-docker/volumes/app/mattermost/data -> maps to /mattermost/data
  + ~/mattermost-docker /volumes/db/var/lib/postgresql/data -> maps to /var/lib/postgresql/data

Production Only -> There is also a volume for https certificates (See Http Certificates)

* ~/certbot-docker/.lego/certificates -> maps to /srv/certs

Staging Only -> There is a different volume for http certificates

* + ~/mattermost-docker /volumes/web/cert/ -> maps to /cert

## Backup/Restoring:

Backup/Restore scripts are located at: <https://gitlab.op-bit.nz/bit-platform/BIT-Useful-Scripts/tree/master/Backup%20&%20Restore%20Scripts>

**All the downloader scripts require the bit-platform-dev.pem**

The backup server contains a samba share to ease the access to the backups and allows the staging machines to copy the latest backup to restore.

*Note: The samba share user only has read access to the folders.*

The Below table shows the Backup and Download Frequency of each server/service

|  |  |  |  |
| --- | --- | --- | --- |
| **Server** | **Backup Freq** | **Download Freq** | **Backup Files** |
| **Gitlab** | Daily 2am | Mon,Wed,Fri,Sat 2:30am | 2 .tar’s   1. Gitlab Config 2. Gitlab Data (Repos,DB, etc) |
| **Mattermost** | Daily 2am | Mon,Wed,Fri,Sat 2:30am | 3 .tar’s   1. Config 2. Data (Uploaded Files etc.) 3. Db (Users, convo’s, etc. |

## HTTPS Certificate Renewal/Issuing:

## Production:

As we use Lets Encrypt to provide us with HTTPS certificates, these need to be renewed every 3 months to ease the process of renewal there is a Docker Container that will request certificates and deal with the domain verification (as we can’t use http/https verification due to the services running on those ports i.e. GitLab).

<https://gitlab.com/op-bit-platform/certbot-docker>

## Staging:

TODO