DevSecCon | SINGAPORE 22-23 FEB 2018

LAB - REMOVE DEVELOPERS' SHAMEFUL SECRETS

OR SIMPLY REMOVE SHAMEFUL DEVELOPERS...

BY Fabian Lim

Join the conversation #DevSecCon

REMOVE DEVELOPERS’ SHAMEFUL SECRETS

Lab Slides

Fair Warning

Although minimal, my scripts are written mainly in MacOS.

For security concerns and compatibility, you might consider executing them in a Linux VM.

The permissions set on the services trust that developers (you) are trustworthy and responsible, with the ability to change certain settings.

Please exercise caution while making changes and limit those changes to your own environment in order to have a conducive learning environment.

Lab Audience and Objective

- Developers - who want to implement and fix the problem
- Managers - who wants the problem to be fixed but don’t know how
- Compliance / Auditors - who wants to see how a problem can be fixed

...

Remove and rotate secrets

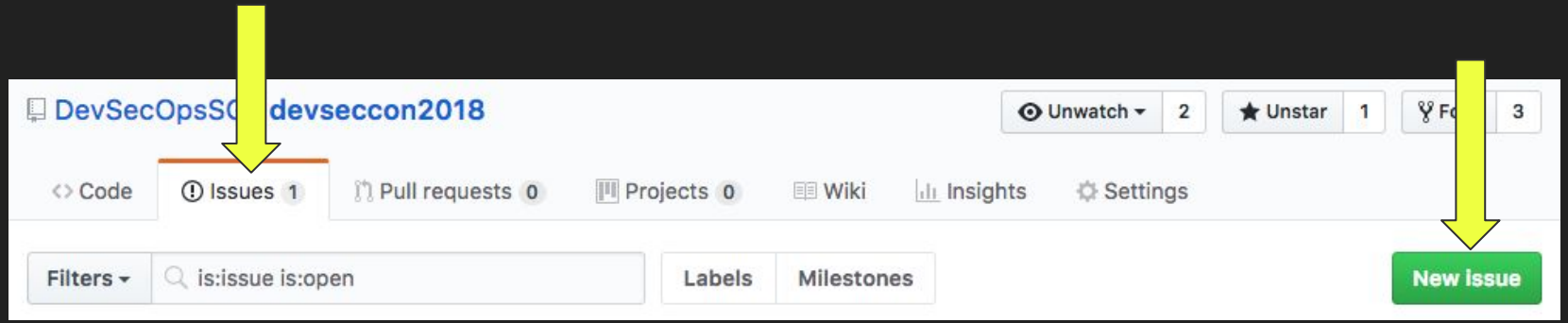
Use a secret management server

Integration with Jenkins

Lab 1 - Secrets Checked In

Setup / Login GitHub

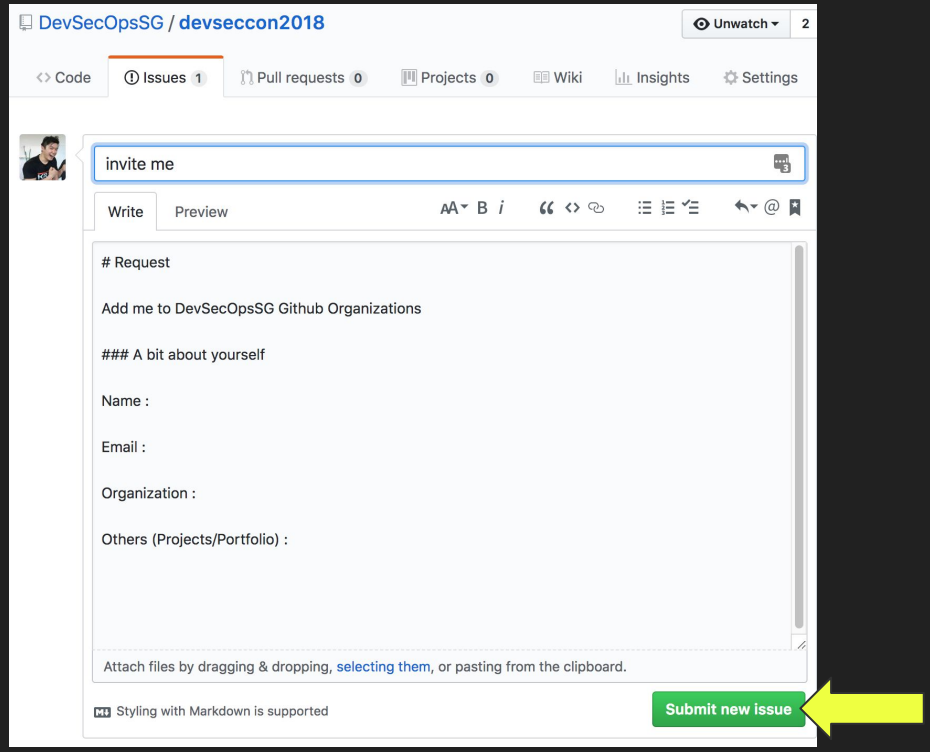
- Login to your GitHub account
- Click on “Issues”
- Click “New issue” to create an issue in the devseccon2018 project so I know your username and I can invite you now!



Setup / Login GitHub

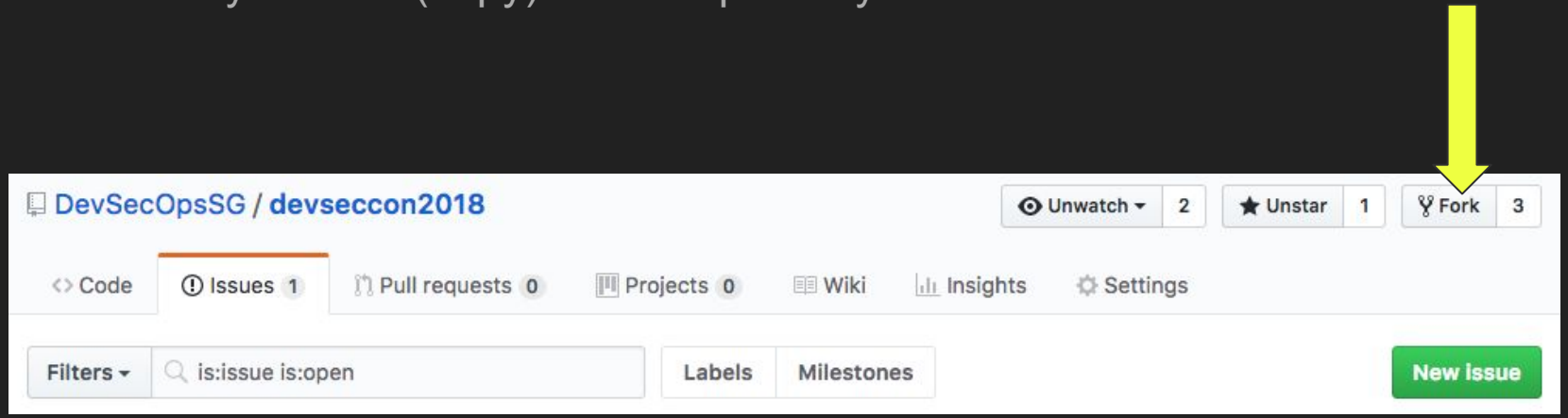
- Fill in some description about yourself and Click “Submit new issue”

- You will receive an email to join
- membership
- to https://github.com/DevSecOpsSG



Setup / Login GitHub

- Go to https://github.com/DevSecOpsSG/devseccon2018.git
- Click “Fork”
- You should be redirected to https://github.com/<username>/devseccon2018.git
- This is your fork (copy) of the repository



Setup / Login GitHub

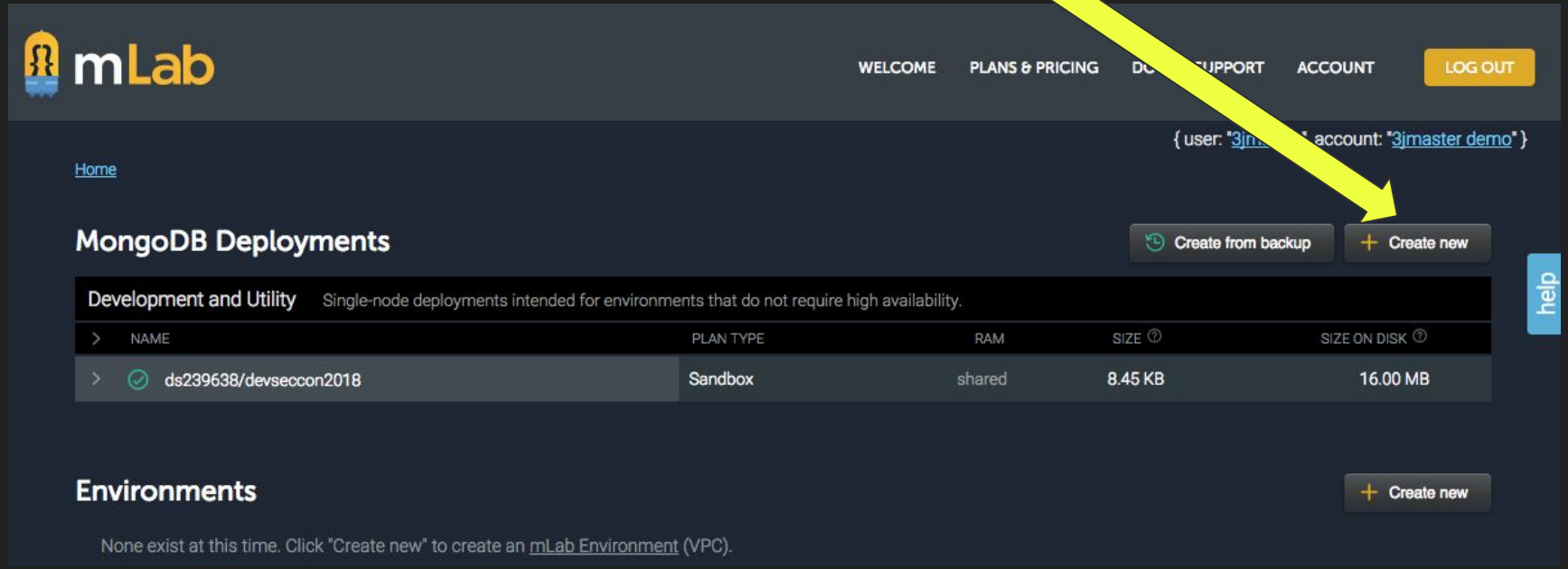
- Open a terminal and run to clone code to your local machine:

git clone https://github.com/<username>/devseccon2018.git

- You now have a local version of the code

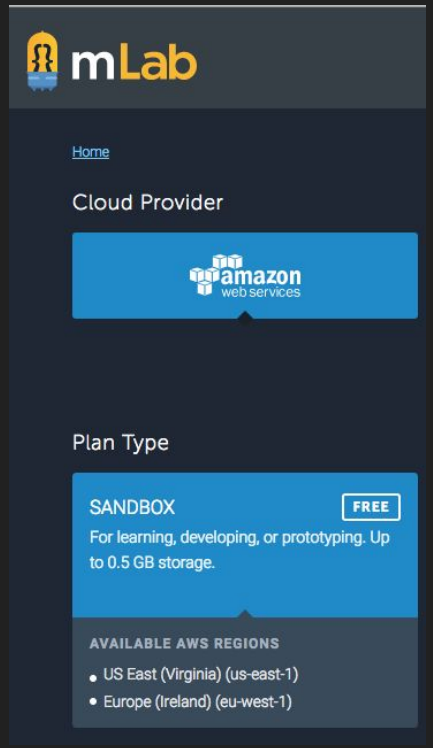
Setup mlab (MongoDB)

- Sign up an account at https://mlab.com
- Click “Create new” for MongoDB Deployments



Setup mlab (MongoDB)

- Choose any Cloud Provider (doesn't matter)
- Choose Plan Type SANDBOX (FREE)
- Click "Continue"



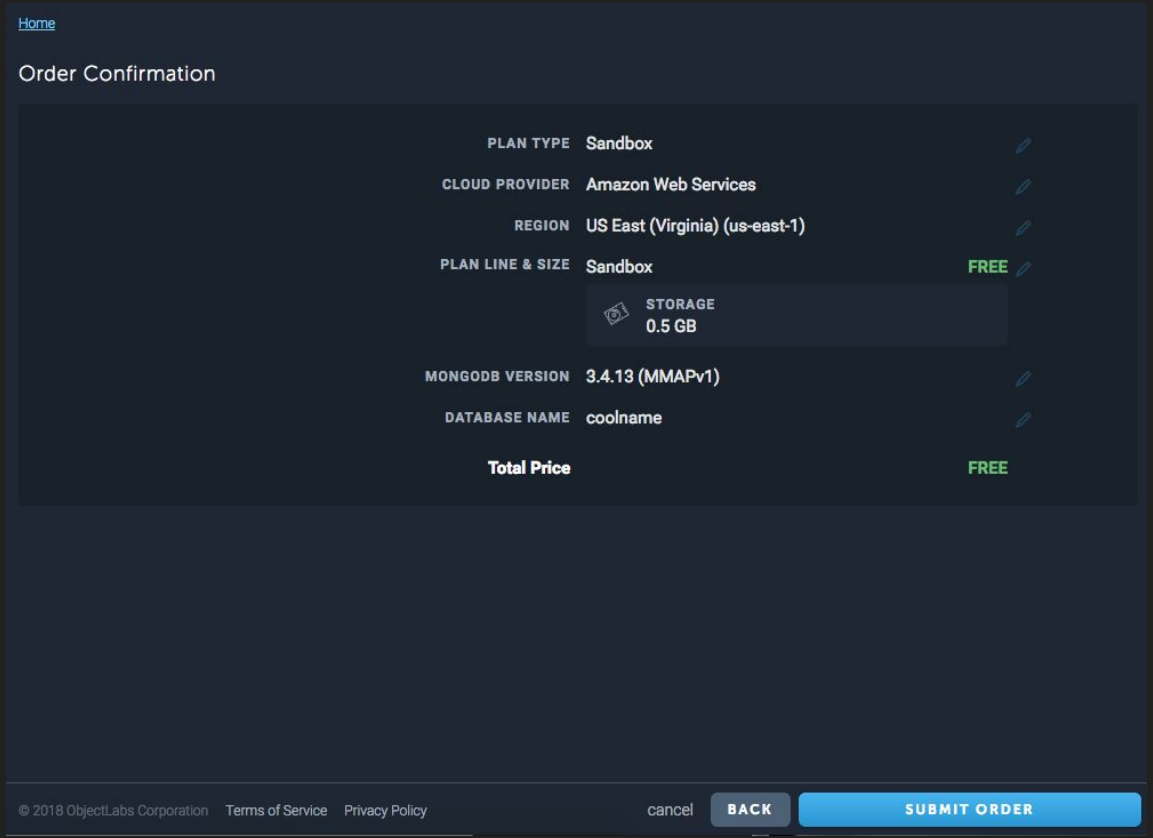
- Select a region (doesn't matter)
- Click "Continue"

Setup mlab (MongoDB)

- Give your database a cool name (doesn’t matter)
- Click “Continue”

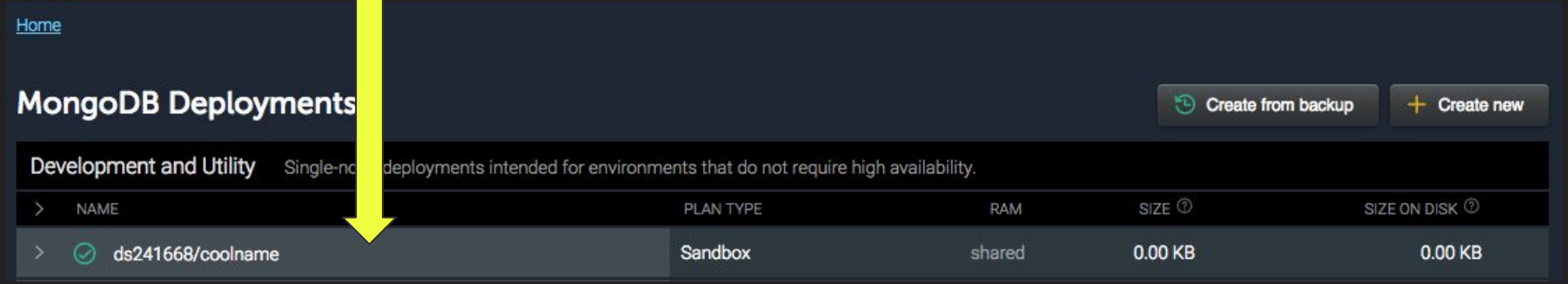
Setup mlab (MongoDB)

- Check your final order
- Click “SUBMIT ORDER”



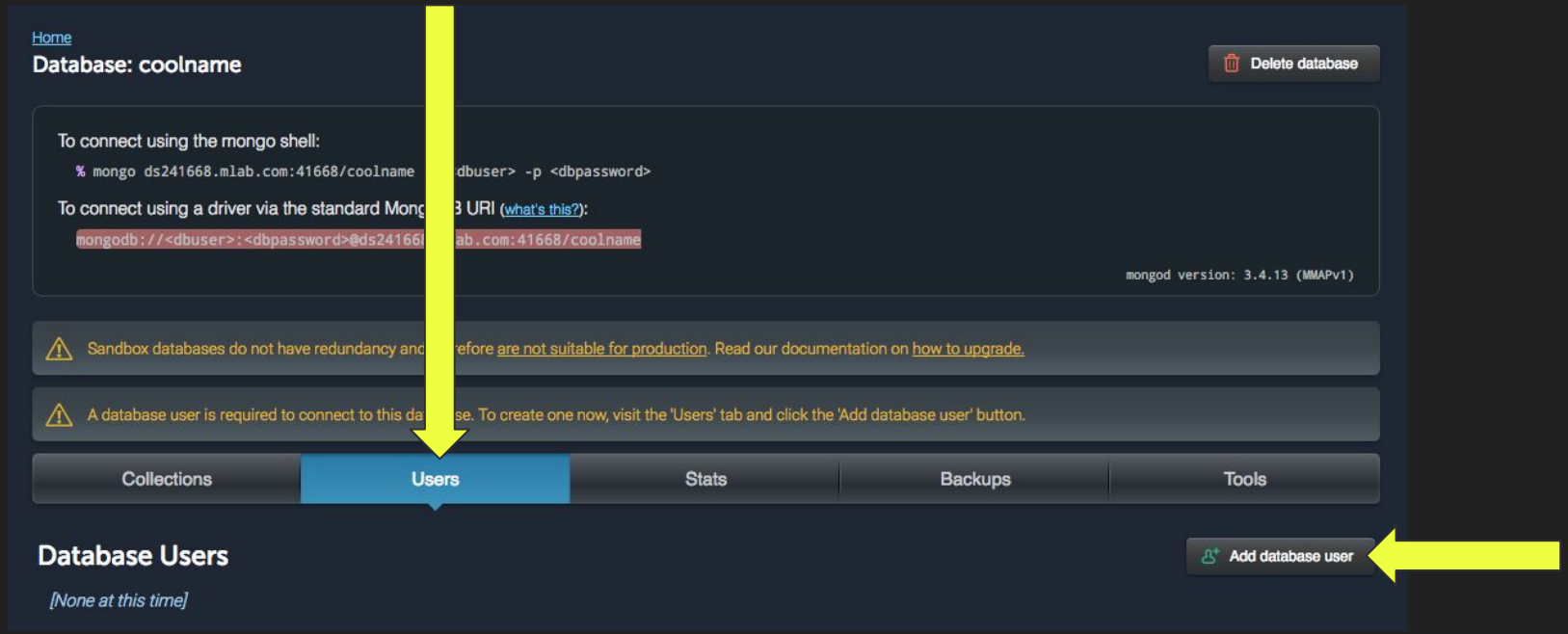
Setup mlab (MongoDB)

- Click on the new database



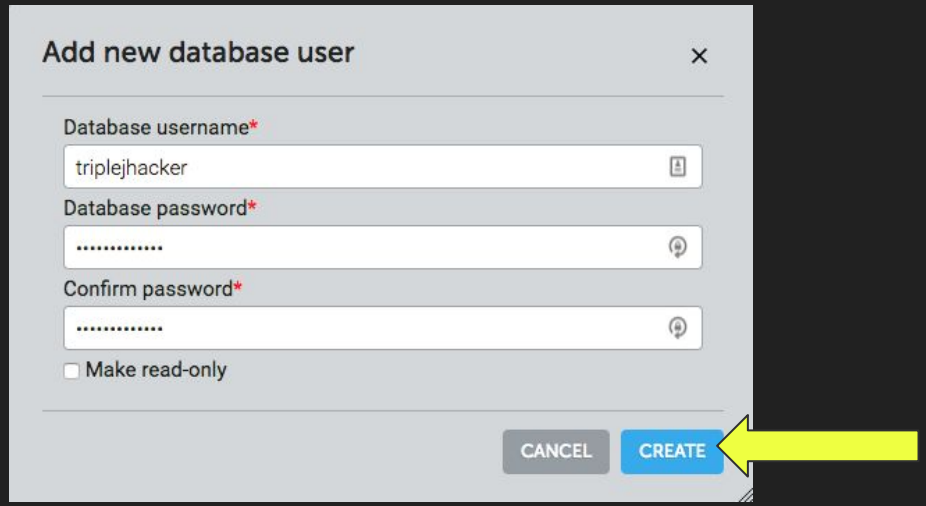
Setup mlab (MongoDB)

- Click on “Users”
- Click on “Add database user”



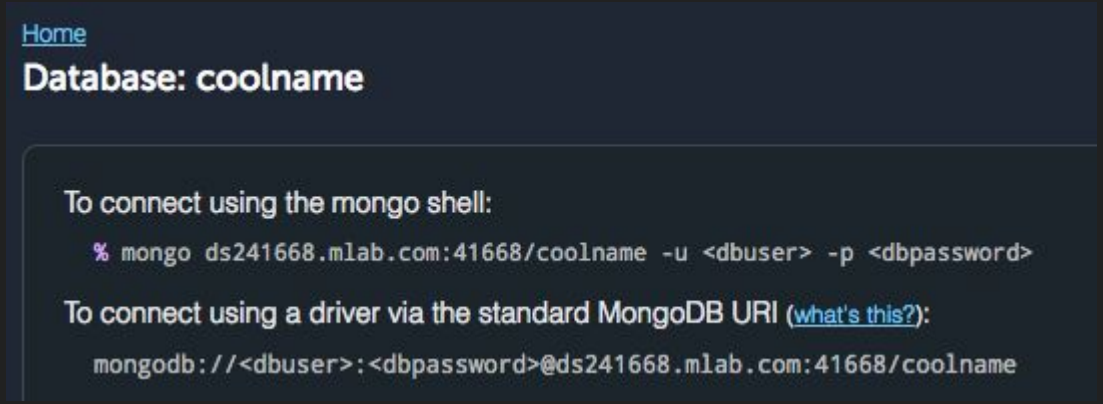
Setup mlab (MongoDB)

- Enter a username and password (don't use anything personal or sensitive)
  - To avoid syntax error in the later steps, do not use the “@” symbol at all; you can use it but you must encode the character in the script later
- Leave “Make read-only” box unchecked
- Click “Create”



Setup mlab (MongoDB)

From the database page, construct your mlab mongodb instance URL (this contains secret) replacing <dbuser> and <dbpassword> that you previously entered.



In my example, mine is:

```
mongodb://triplejhacker:<dbpassword>@ds241668.mlab.com:41668/coolname
```

Paste mlab URL as MONGODB_CREDENTIALS

start_app_server.sh — ~/Dev/devseccon2018

Project

- devseccon2018
  - .git
  - .sonarlint
  - lab
  - slides
  - views
  - .DS_Store
  - .gitignore
  - app.js
  - Dockerfile
  - jenkins.build.sh
  - package.json
  - README.md
  - start_app_server.sh

start_app_server.sh

```
1  #!/bin/sh
2  # README
3
4  # READ SECRET!
5  export MONGODB_CREDENTIALS=mongodb://test:testtest@ds239638.mlab.com:39638/devseccon2018
6
7  # finally, run the app in dev mode...
8  npm run dev
9
```

Push changes to your fork

```
git add start_app_server.sh

git commit -m ‘added mongo credentials’

git push origin master
```

Yes, check your secret into the code repository :P

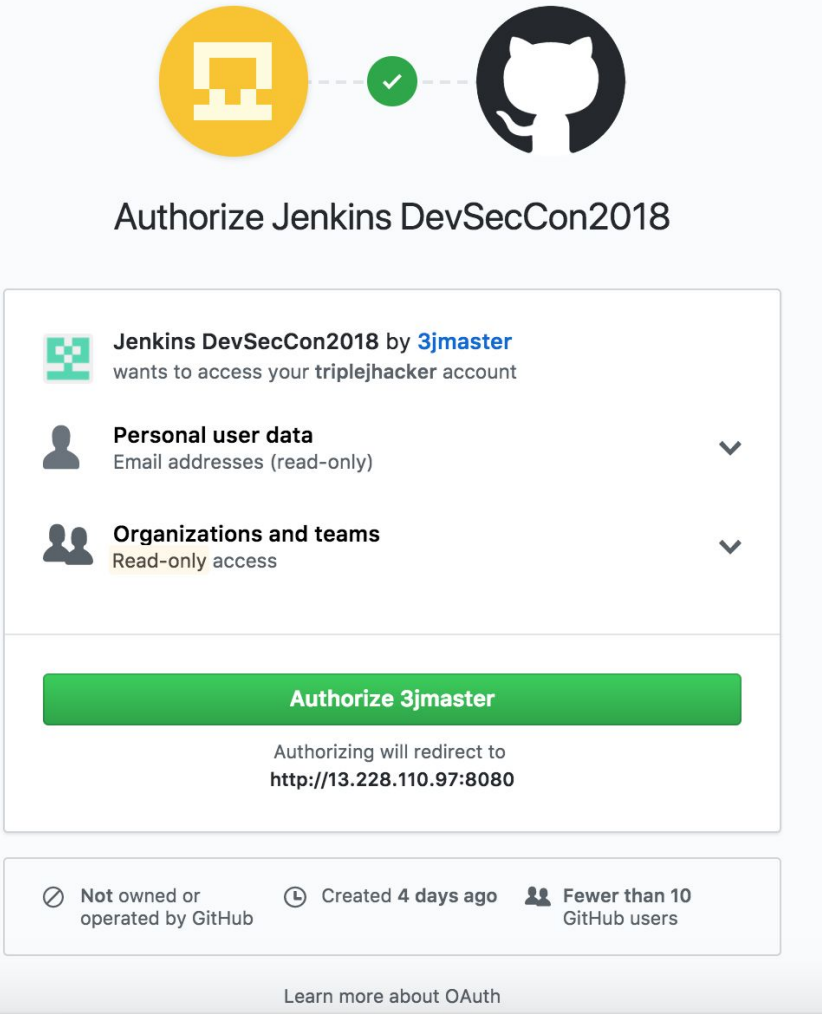
We will remove and rotate it later, don’t worry.

Access to Jenkins

Go to http://13.228.110.97:8080

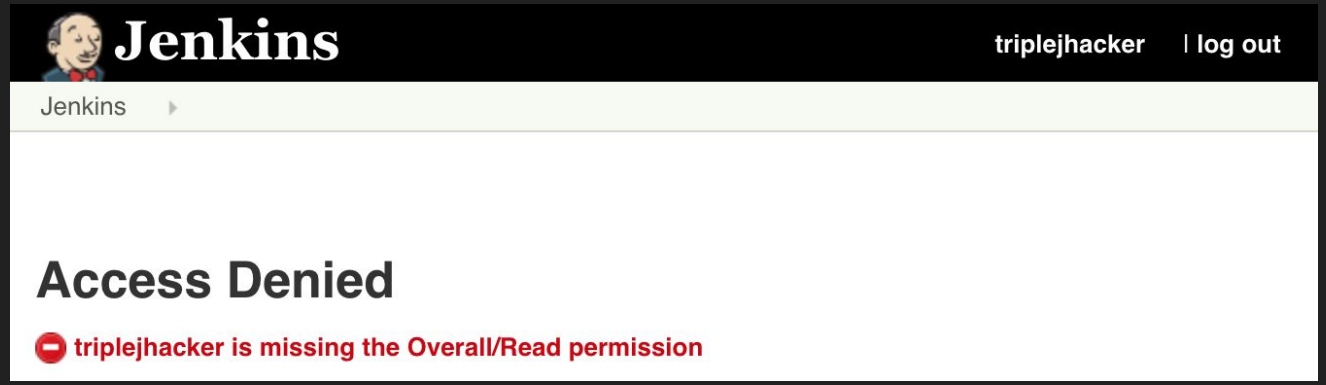
Membership in https://github.com/DevSecOpsSG allows access to this Jenkins server.

Click “Authorize 3jmaster” (That’s me)



Denied Access to Jenkins

If you see this



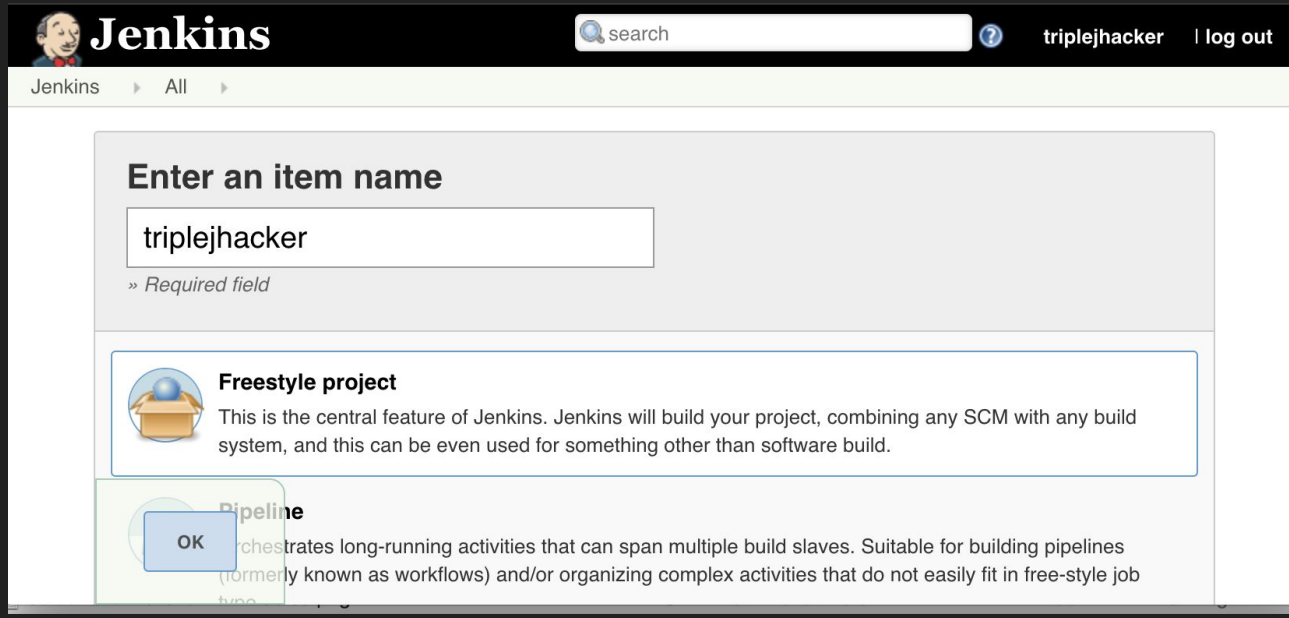
Please perform steps in slide 5 and 6 to get access to the Github organization membership

Build and Run with Jenkins

Click “New Item”

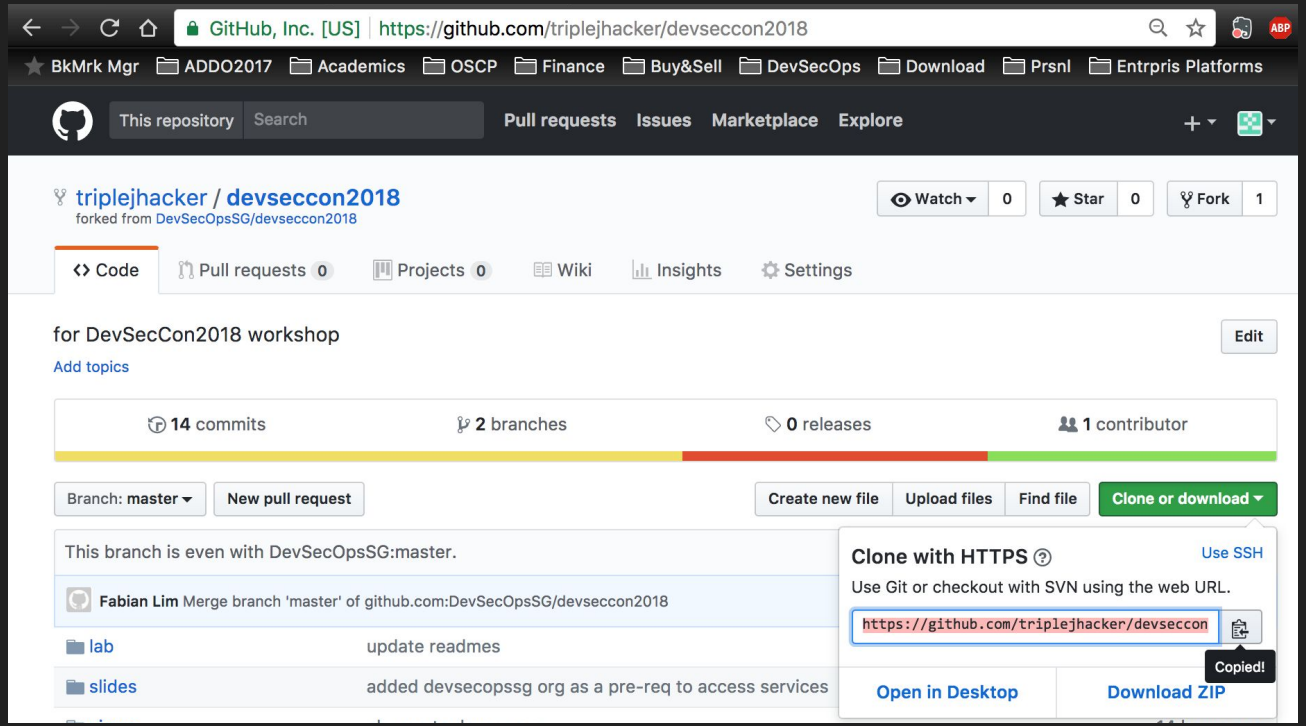
Build and Run with Jenkins

- Enter your username or any name as a project name
- Choose “Freestyle Project”
- Click “OK”



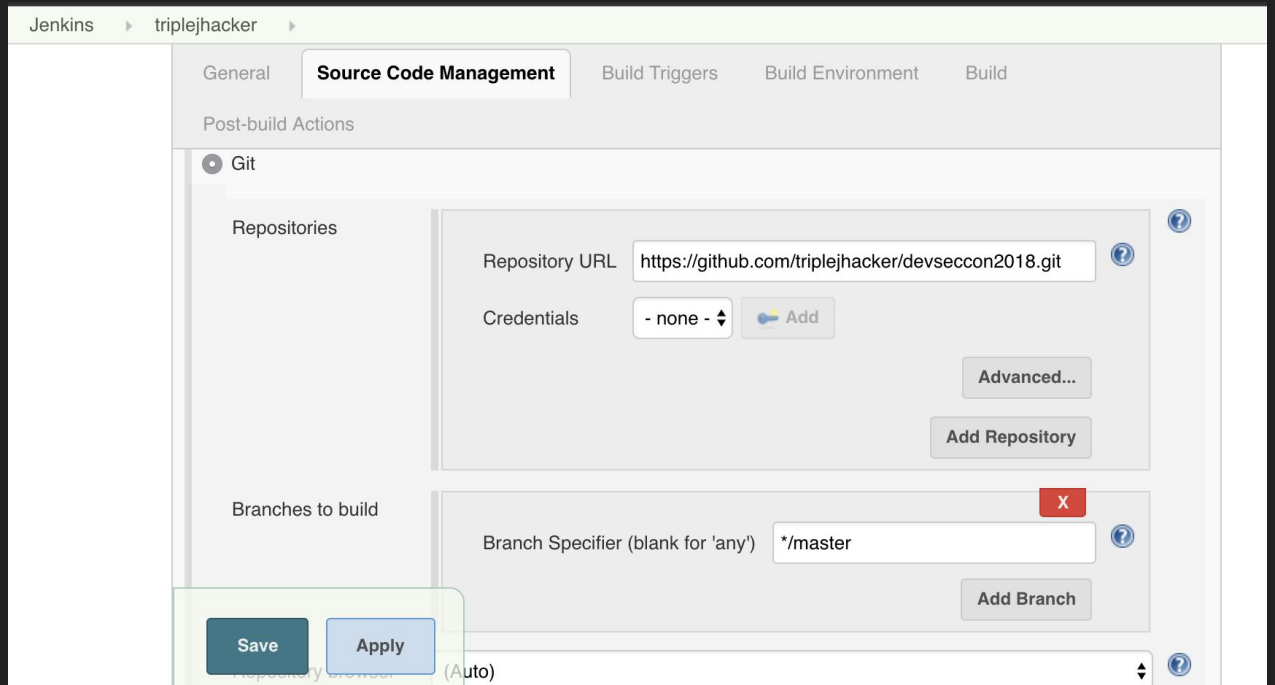
Build and Run with Jenkins

- Go to your fork of the repository and copy the URL from “Clone or download”



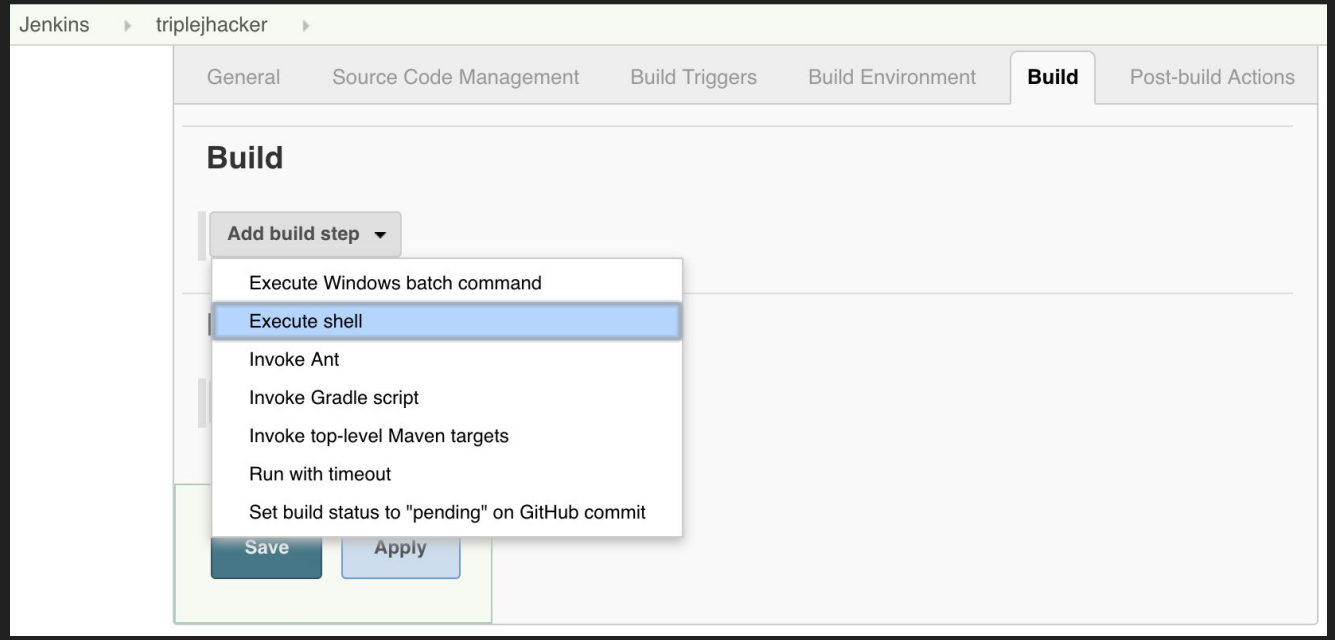
Build and Run with Jenkins

- Paste it in Jenkins under “Configure”, under Source Code Management -> Git -> Repositories -> URL



Build and Run with Jenkins

- Go to: Build -> Add Build Step -> Execute Shell

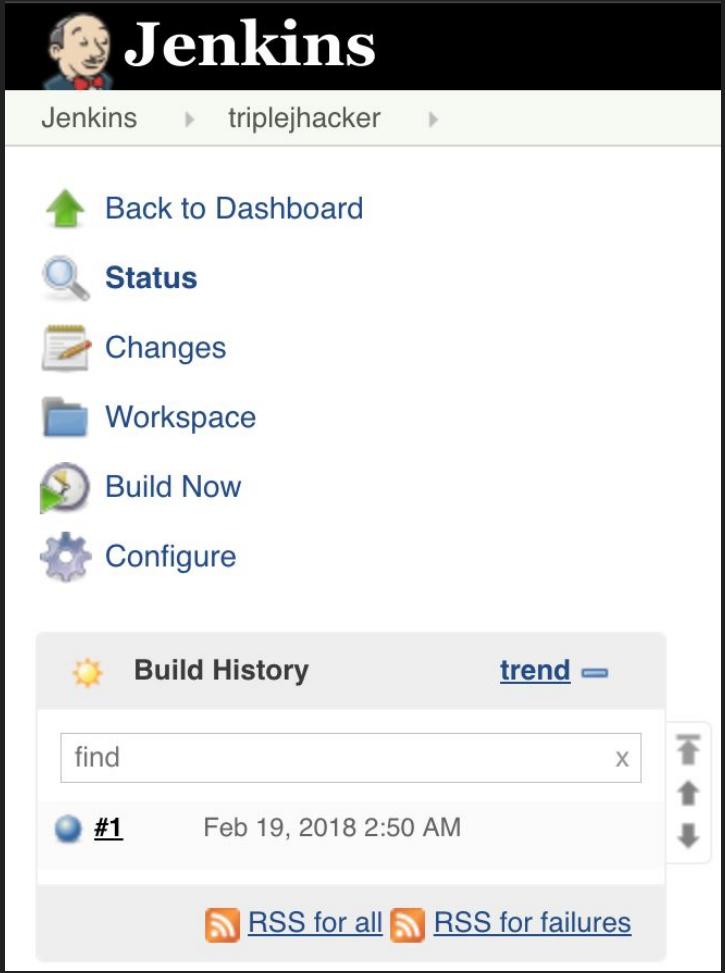


Build and Run with Jenkins

- Copy and paste the contents of jenkins.build.sh from the code repository
- Replace IMG_NAME (with your username), CTNR_NAME (with your username), PORT (with a random number between 9000 and 9999)
- ***Remember this PORT value, you will need to append it to the URL later***
- Click “Save”

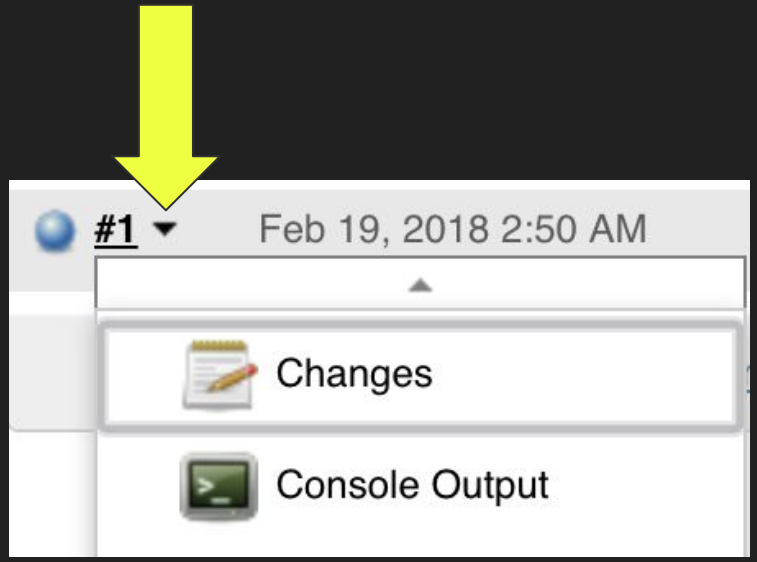
Build and Run with Jenkins

- Click “Build Now”
- Under “Build History”, there should be a build number like “#1”



Build and Run with Jenkins

- Click on the arrow beside the build number (shown here)
- Click on “Console Output” to show the logs from the build



Build and Run with Jenkins

- If all goes well, it should look something like this ending with “**Finished: SUCCESS**”

- *If you encounter error with container name, the CTNR_NAME that was already been used. So, change the value of CTNR_NAME, save project and re-run “Build Now”.*

- *If you encounter error with port number, a PORT that was already been used, so change the value of PORT, save project and re-run “Build Now”.*

Jenkins ▸ triplejhacker ▸ #1

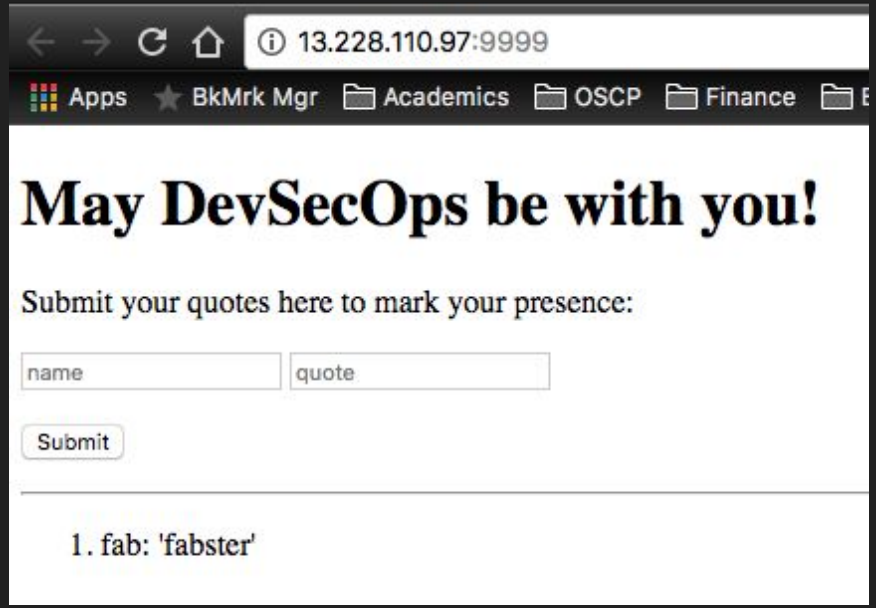
Console Output

```
Started by user triplejhacker
Building in workspace /var/lib/jenkins/workspace/triplejhacker
Cloning the remote Git repository
Cloning repository https://github.com/triplejhacker/devseccon2018.git
 > git init /var/lib/jenkins/workspace/triplejhacker # timeout=10
Fetching upstream changes from https://github.com/triplejhacker/devseccon2018.git
 > git --version # timeout=10
 > git fetch --tags --progress https://github.com/triplejhacker/devseccon2018.git
+refs/heads/*:refs/remotes/origin/*
 > git config remote.origin.url https://github.com/triplejhacker/devseccon2018.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/triplejhacker/devseccon2018.git # timeout=10
Fetching upstream changes from https://github.com/triplejhacker/devseccon2018.git
 > git fetch --tags --progress https://github.com/triplejhacker/devseccon2018.git
+refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 5518845d1c9d2a08f96dfdfb7e87c060efb7d652 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 5518845d1c9d2a08f96dfdfb7e87c060efb7d652
Commit message: "Merge branch 'master' of github.com:DevSecOpsSG/devseccon2018"
First time build. Skipping changelog.
[triplejhacker] $ /bin/sh -xe /tmp/jenkins8763337294582057440.sh
+ export IMG_NAME=triplejhacker/devseccon2018
+ IMG_NAME=triplejhacker/devseccon2018
+ export CTNR_NAME=triplejhacker
+ CTNR_NAME=triplejhacker
+ export PORT=9999
+ PORT=9999
+ sudo docker kill triplejhacker
Error response from daemon: Cannot kill container: triplejhacker: No such container: triplejhacker
+ true
+ sudo docker rm triplejhacker
Error response from daemon: No such container: triplejhacker
+ true
+ sudo docker build -t triplejhacker/devseccon2018 .
Sending build context to Docker daemon  19.65MB

Step 1/8 : FROM node:alpine
 ---> b7e15c83cdaf
Step 2/8 : WORKDIR /usr/src
 ---> Using cache
 ---> fdead47f886f
Step 3/8 : COPY package.json .
 ---> Using cache
 ---> 590500544dea
Step 4/8 : RUN npm install
 ---> Using cache
 ---> a89ace4cc5ad
Step 5/8 : COPY . .
 ---> 10612a60f4b7
Step 6/8 : EXPOSE 3000
 ---> Running in 7875e2b6c769
 ---> 176a023dda0e
Removing intermediate container 7875e2b6c769
Step 7/8 : COPY ./start_app_server.sh /usr/local/bin/start_app_server.sh
 ---> 441d495a728c
Step 8/8 : CMD start_app_server.sh
 ---> Running in c6562ca0c15b
 ---> d7f3d7cd1d1e
Removing intermediate container c6562ca0c15b
Successfully built d7f3d7cd1d1e
Successfully tagged triplejhacker/devseccon2018:latest
+ sudo docker run --network=isolated_nw -d -p 9999:3000 --name triplejhacker
triplejhacker/devseccon2018
a559e97ee9cd7b2d711a26d6f874f363f311005ebc1533f161a5e1fa239af5f5
Finished: SUCCESS
```

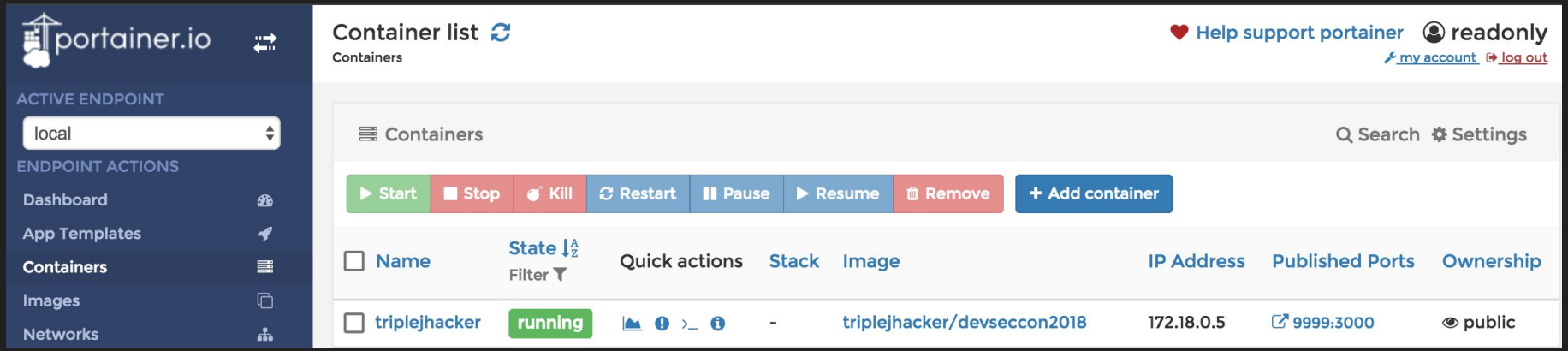
Access your deployed app

- Append the port number you specified in jenkins to http://13.228.110.97:<specified_port> and go to this URL in your browser
- In my example, I go to http://13.228.110.97:9999
- A simple app should display
- Interact with the app by adding quotes
- These quotes are stored in your mongodb in mlab. You can go back to mlab and check the changes in database



Debug your deployed app

- Docker UI is at http://13.228.110.97:8100
- Login with username and password “readonly” to view the state and logs of containers



Back to Slide

https://docs.google.com/presentation/d/1jW0pPXheS2aZqsXvfPATQLbY5sDRyGVpuNpswS5Zv4I/edit#slide=id.g31d5e508b0_0_590

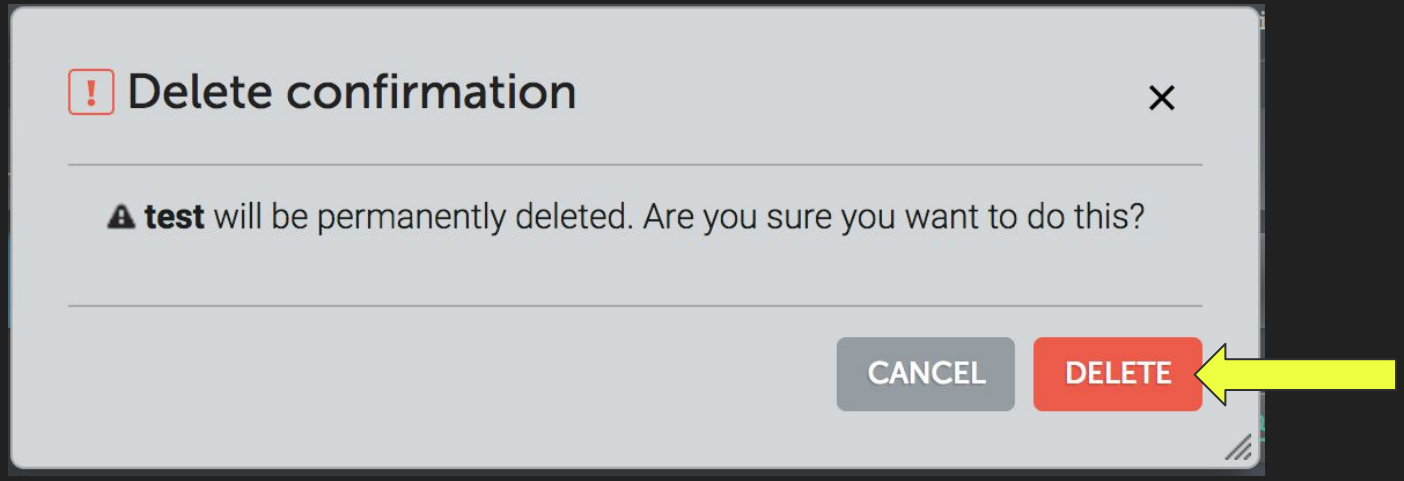
Solution 1 - Remove secret from App, secret stays in Jenkins

Rotate mlab (MongoDB) credentials

- Back on the mlab page,
- Click on “Users”
- Click on the trash bin icon to delete the user

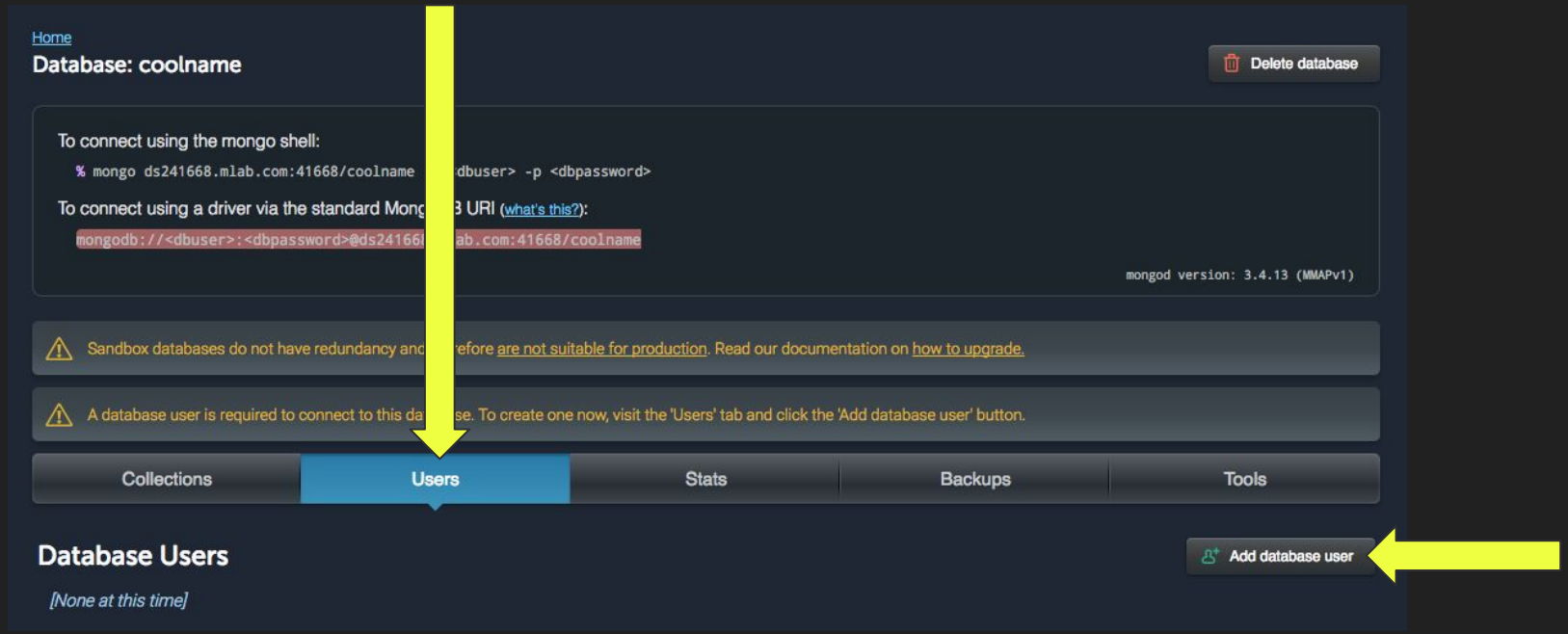
Rotate mlab (MongoDB) credentials

- Go ahead and Click on “DELETE”



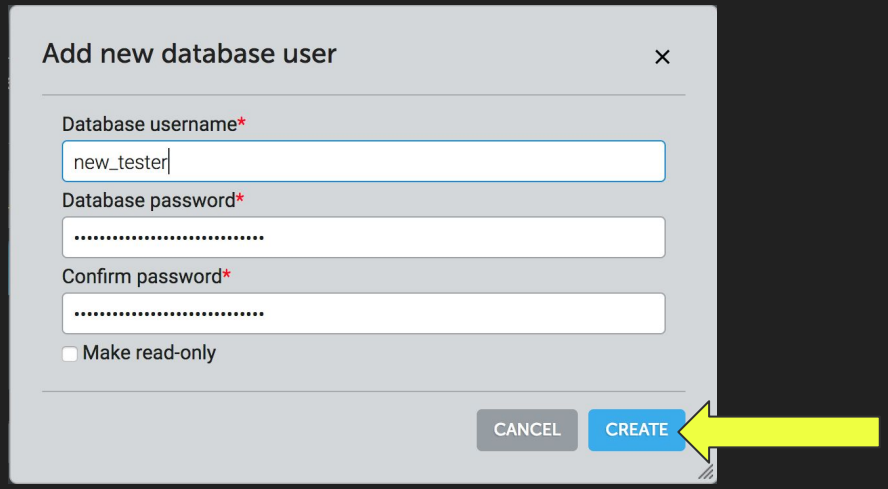
Rotate mlab (MongoDB) credentials

- Click on “Add database user”



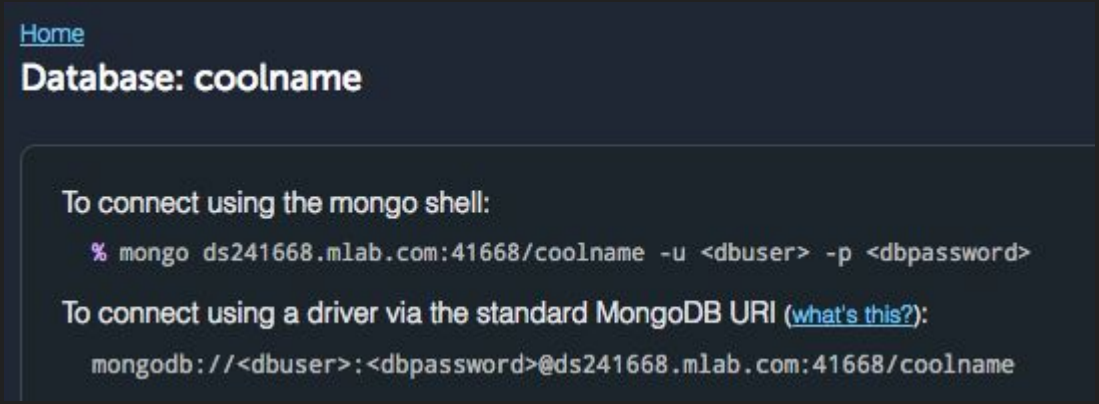
Rotate mlab (MongoDB) credentials

- Enter a NEW username and password (don't use anything personal or sensitive)
  - To avoid syntax error in the later steps, do not use the “@” symbol at all; you can use it but you must encode the character in the script later
- Leave “Make read-only” box unchecked
- Click “Create”



Rotate mlab (MongoDB) credentials

From the database page, construct your mlab mongodb instance URL (this contains secret) replacing <dbuser> and <dbpassword> that you previously entered.

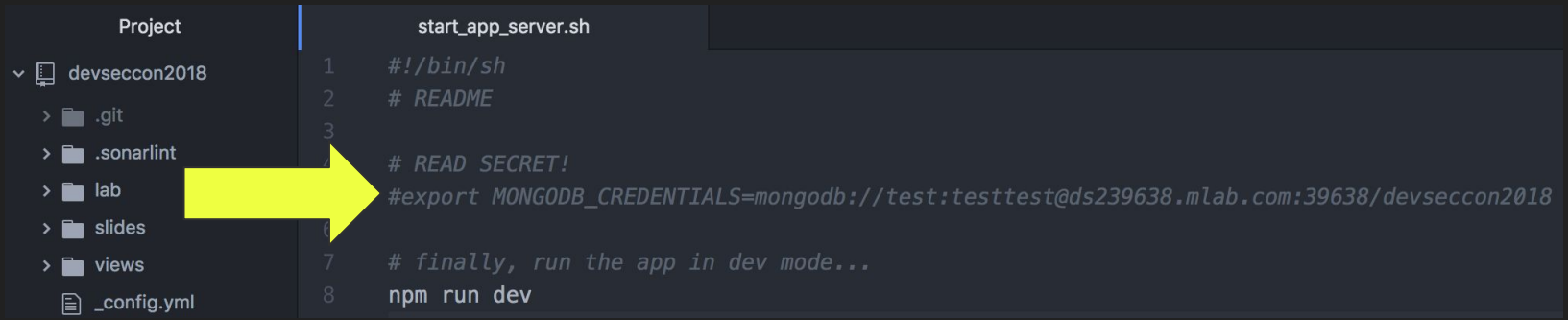


In my example, mine is:

```
mongodb://new_tester:<dbpassword>@ds241668.mlab.com:41668/coolname
```

Remove secret from code repository

From your local machine, delete or comment out the secret mongodb url:



Remove secret from code repository

From your local machine in the directory where the git repository is, run:

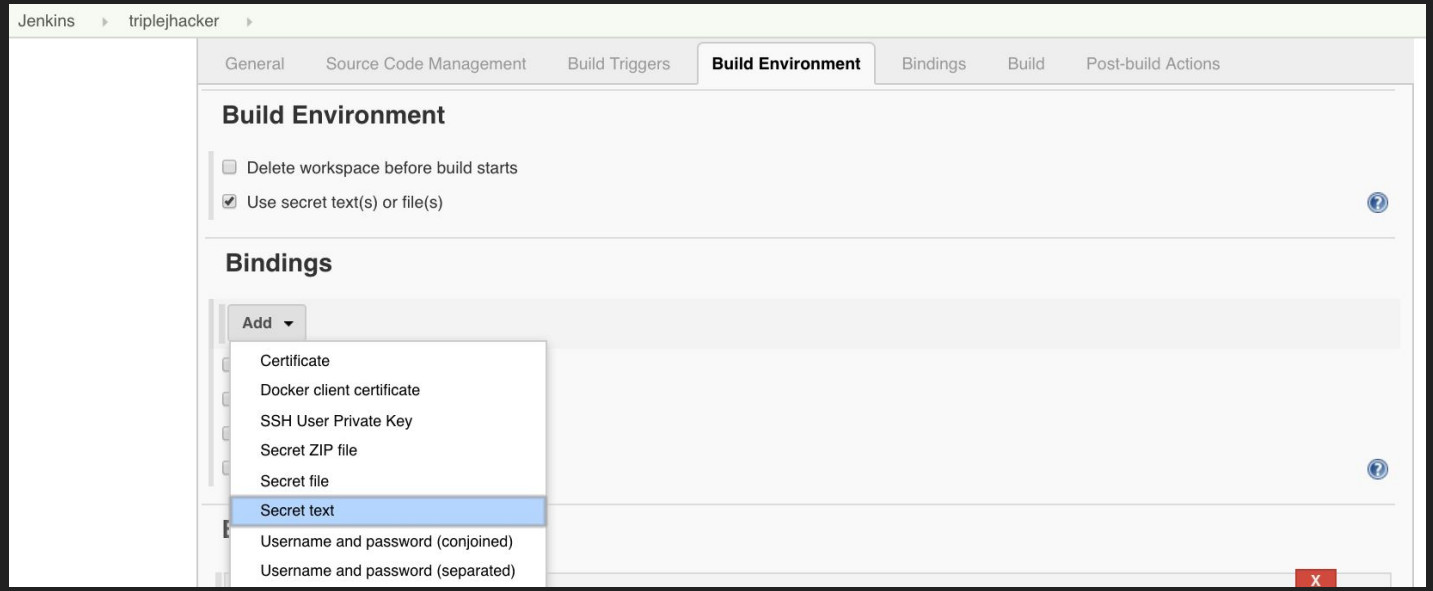
```
git add start_app_server.sh

git commit -m 'removed secret'

git push origin master
```

Store secrets within Jenkins

- Go to your previously created Jenkins project, under “Configure”
- Check “Use secret text(s) file(s)”
- Click “Secret Text”

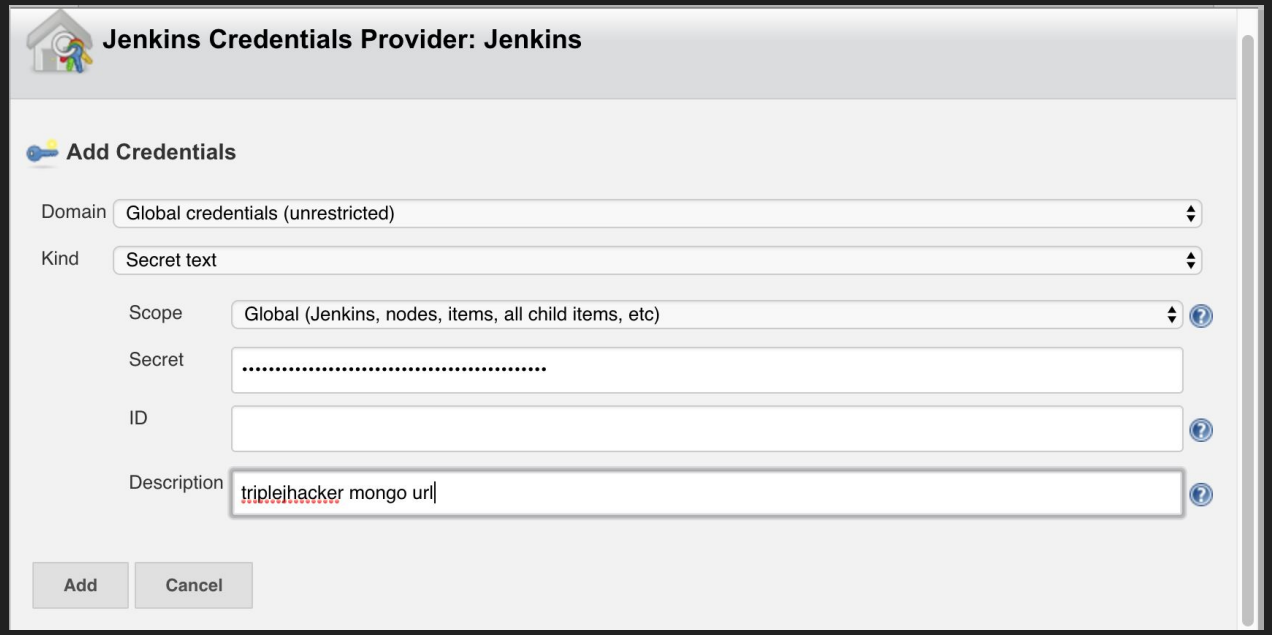


Store secrets within Jenkins

- Fill Variable as “mongodb”
- Add to “Jenkins”

Store secrets within Jenkins

- Choose “Secret text” as kind
- Fill Secret as your mongodb URL (from slide 7)
- Fill Description with your username (for easy identification)
- Click “Add”

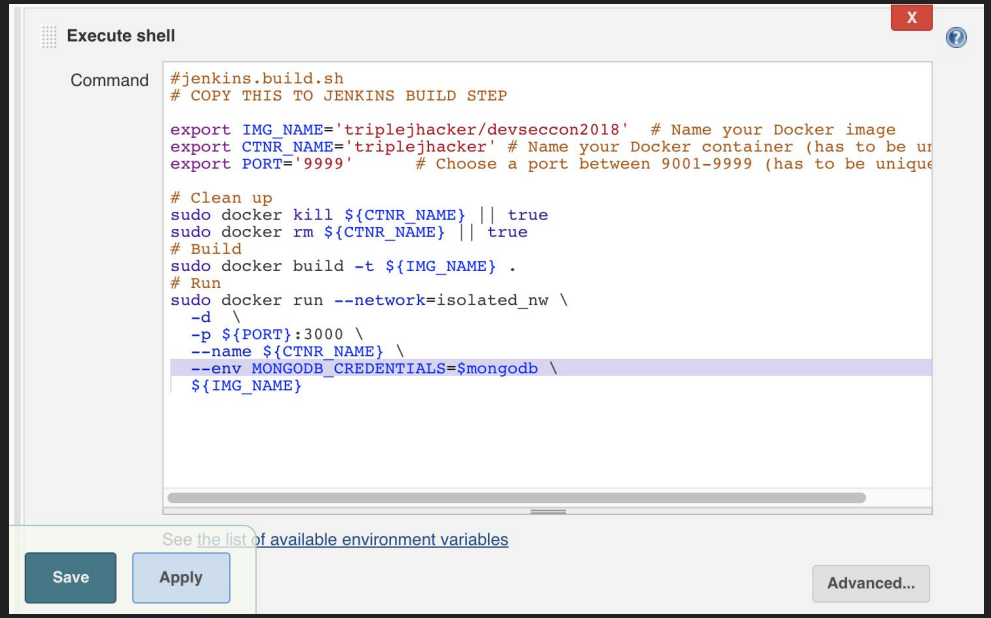


Store secrets within Jenkins

- In the “Execute Shell”, add a line under `docker run`:

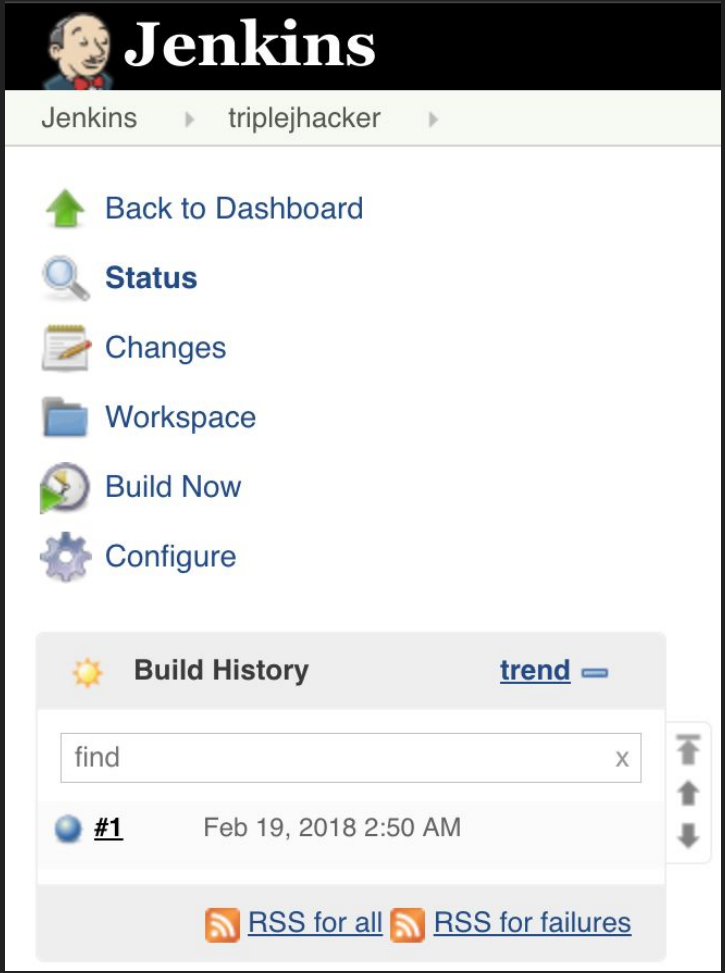
  “`--env MONGODB_CREDENTIALS=$mongodb \`”

MUST add the backslash “\” at the end!



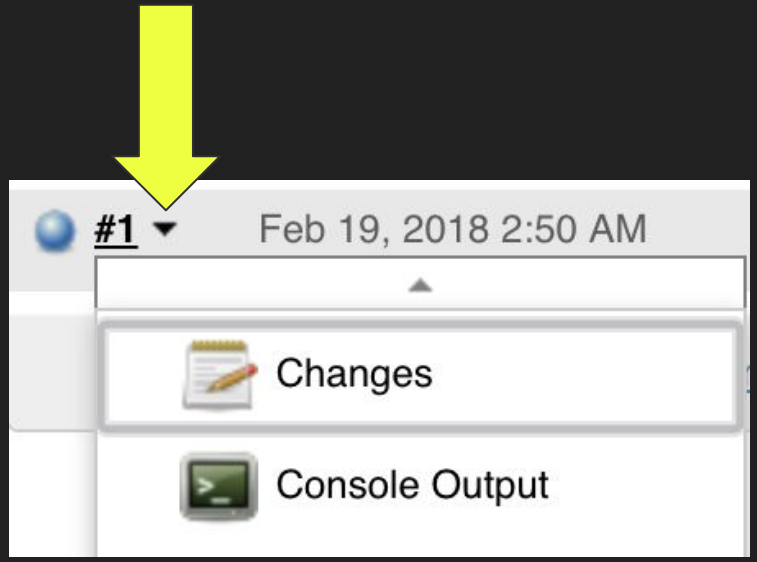
Build and Run with Jenkins

- Click “Build Now”
- Under “Build History”, there should be a build number like “#1” or “#2” or…



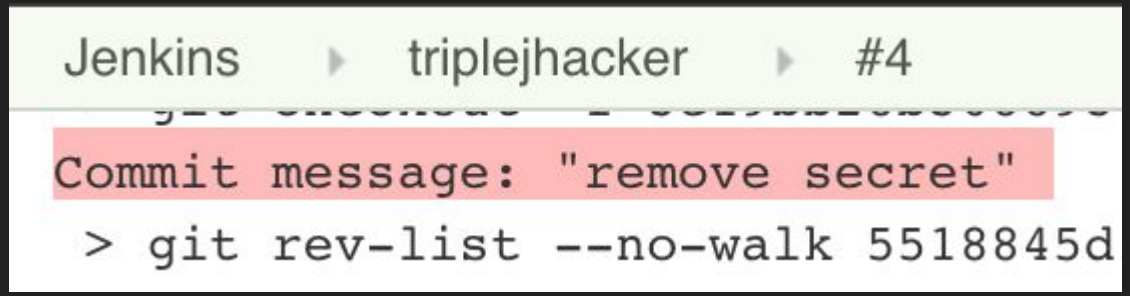
Build and Run with Jenkins

- Click on the arrow beside the build number (shown here)
- Click on “Console Output” to show the logs from the build



Build and Run with Jenkins

- If all goes well, it should be pulling from your latest code commit, check the commit message.
  - It should be the same as the value in “`git commit -m ‘remove secret’` ” ran earlier



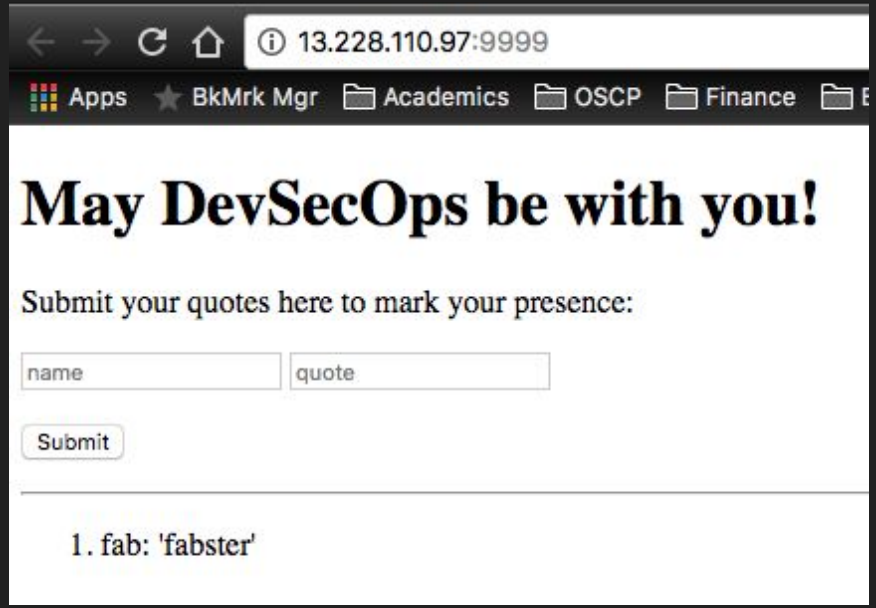
Build and Run with Jenkins

- If all goes well, it should look something like this ending with “**Finished: SUCCESS**”
- Note that the secret is also masked out. Good job Jenkins!

```
+ sudo docker run --network=isolated_nw -d -p 9999:3000 --name triplejhacker --env MONGODB_CREDENTIALS=****
triplejhacker/devseccon2018
2704aa320979f1c615df5db102fac4e0242bc5c6fbabada54c3809bbc4ba08a4
Finished: SUCCESS
```

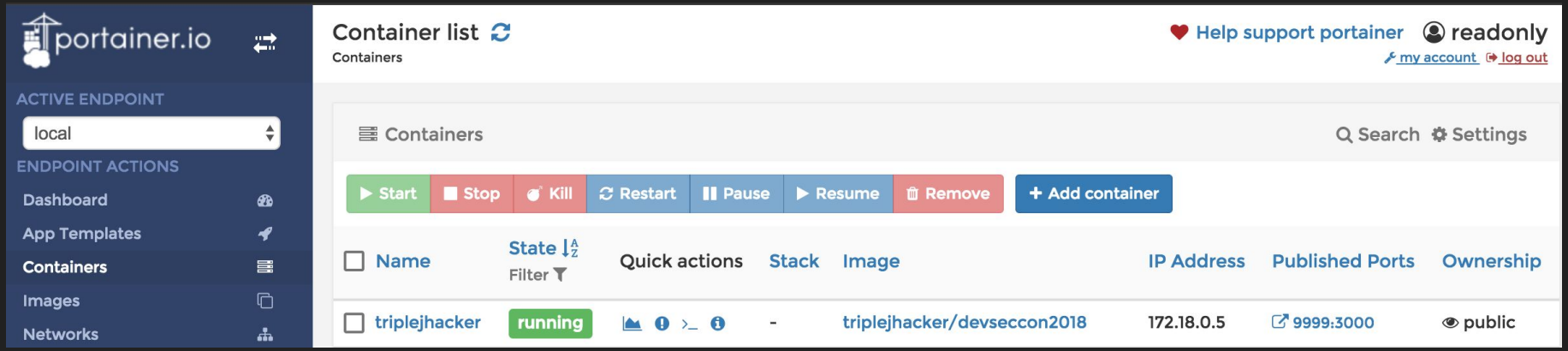
Access your deployed app

- Append the port number you specified in jenkins to http://13.228.110.97:<specified_port> and go to this URL in your browser
- In my example, I go to http://13.228.110.97:9999
- A simple app should display
- Interact with the app by adding quotes
- These quotes are stored in your mongodb in mlab. You can go back to mlab and check the changes in database



Debug your deployed app

- Docker UI is at http://13.228.110.97:8100
- Login with username and password “readonly” to view the state and logs of containers



Congrats!
You have just removed (and rotated) a shameful secret!

But is this good enough?

Solution 2 - Remove Secrets from App and Jenkins, using Vault and its App Role

What is App Role?

https://www.vaultproject.io/docs/auth/approle.html

Access to Vault

- Membership in https://github.com/DevSecOpsSG allows access to Vault
- Generate a GitHub personal access token to login -> Follow Steps 1-9 on: https://help.github.com/articles/creating-a-personal-access-token-for-the-command-line/
- Scopes define the access for personal tokens: Check “**read:org**” only

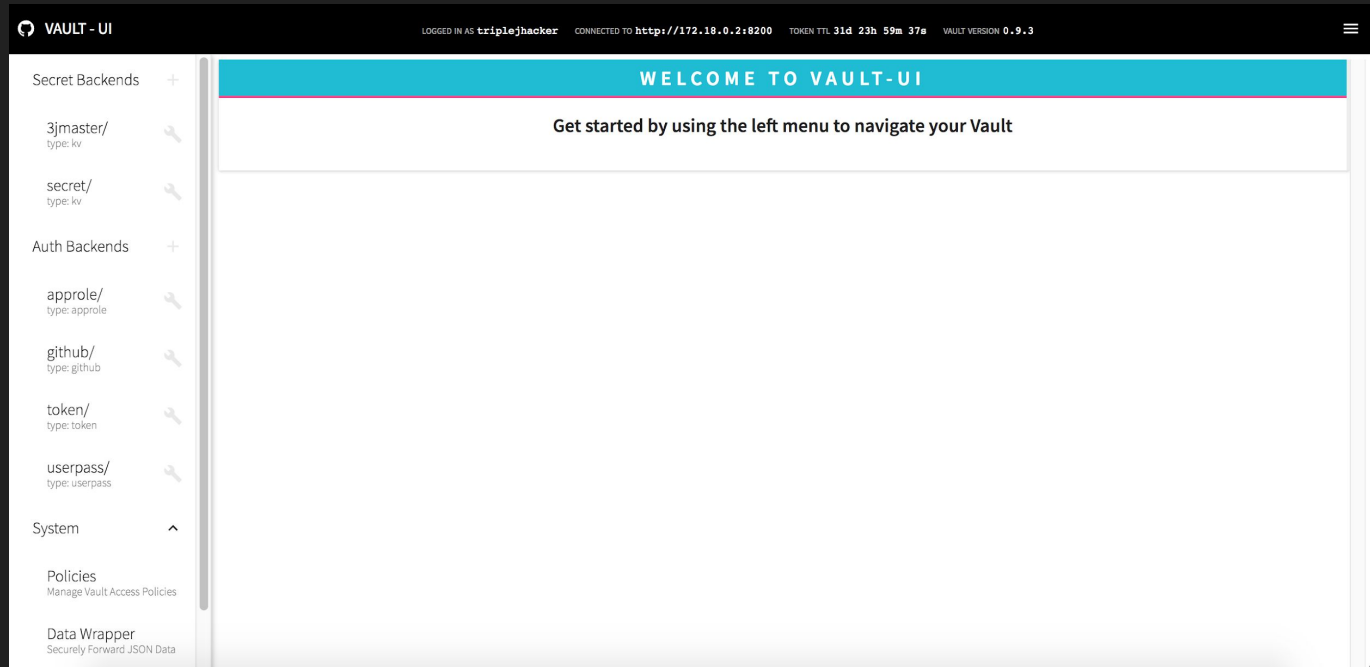
☐ **admin:org**	Full control of orgs and teams
☐ write:org	Read and write org and team membership
☑ read:org	Read org and team membership

Access to Vault (GUI version)

- This is Vault's UI (web container)

Note:
Vault UI
runs on port 8300

While Vault Server
runs on port 8200



Access to Vault (CLI version)

- Replace the GitHub Personal Token with your own and run:

```
$ export VAULT_ADDR=http://13.228.110.97:8200
$ vault auth -method=github token=<$YOUR_GITHUB_PERSONAL_TOKEN>
Successfully authenticated! You are now logged in.
The token below is already saved in the session. You do not
need to "vault auth" again with the token.
token: ************************
token_duration: 2764799
token_policies: [default]
```

Access to Vault (CLI version)

- Replace the username and secret with your own and run:

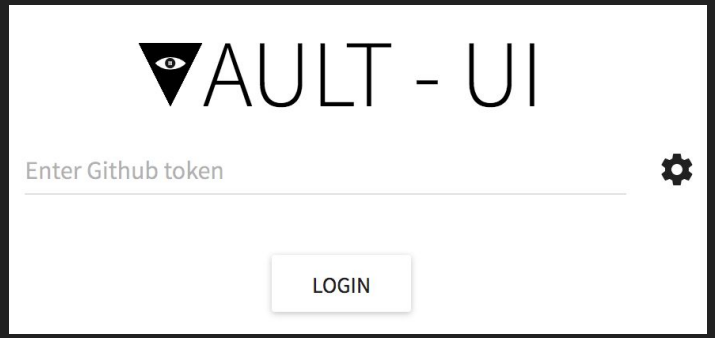
```
$ vault write secret/example/triplejhacker mongodb=<$MONGODB_URL_SECRET>
Success! Data written to: secret/example/triplejhacker
$ vault read secret/example/triplejhacker
Key                 Value
---                 -----
refresh_interval    768h0m0s
mongodb             <$MONGODB_URL_SECRET>
```

This command overwrites any existing values

You can now skip slides for Access to Vault (GUI version)

Access to Vault (GUI version)

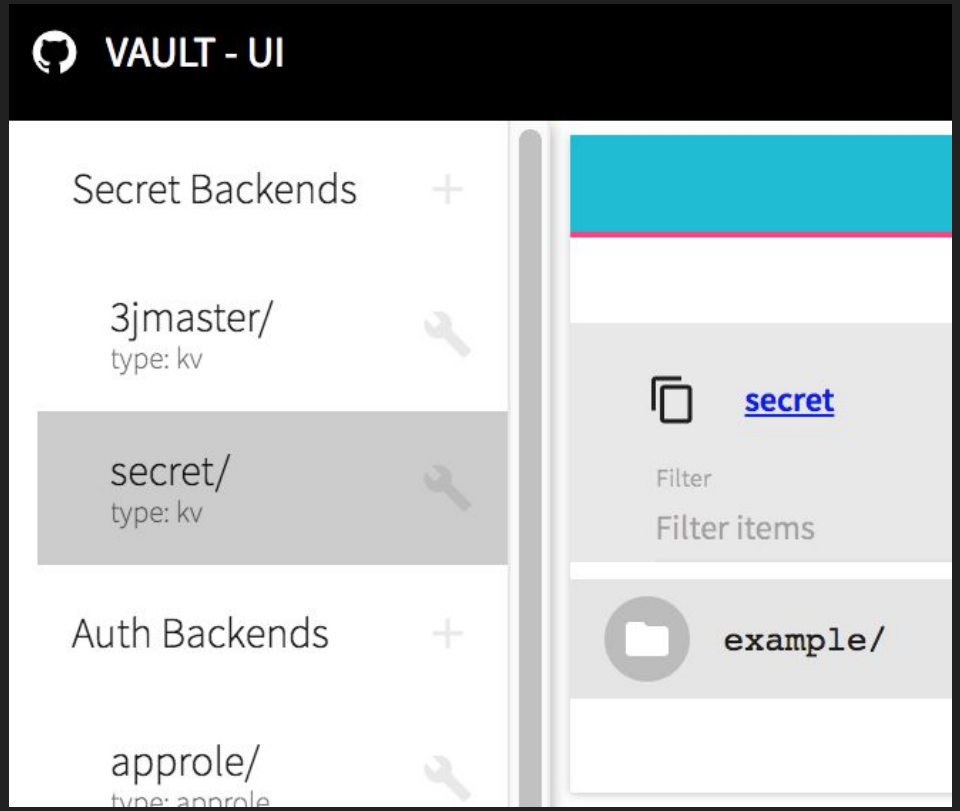
- Go to http://13.228.110.97:8300
- Paste the GitHub Personal Token and login



- If it is not showing this, then Click on settings and choose "GitHub" as Login Method. *Do not change the Vault Server URL*
- Click “OK”
- Paste the GitHub Personal Token and login

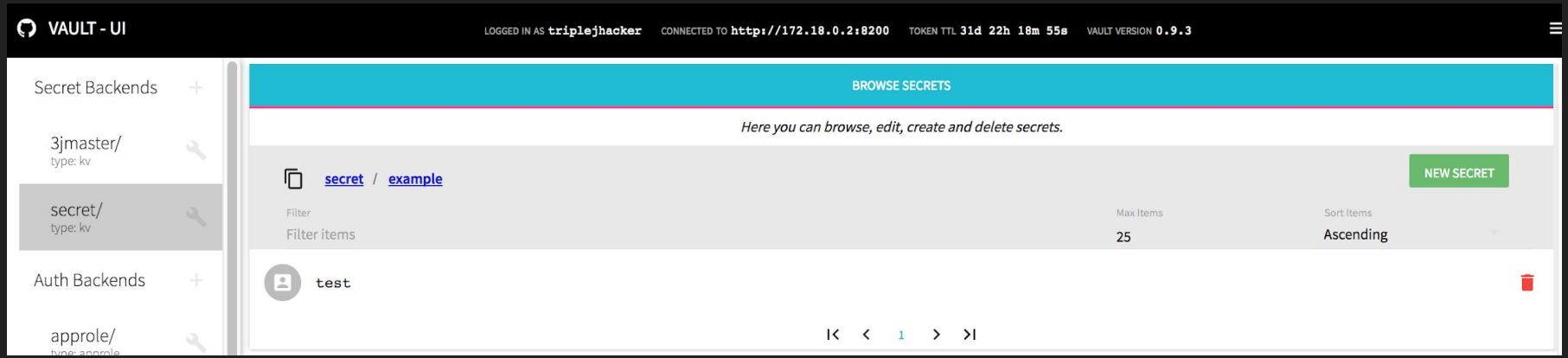
Access to Vault (GUI version)

- Under Secret Backend, click “secret/”
- Click “example/” folder



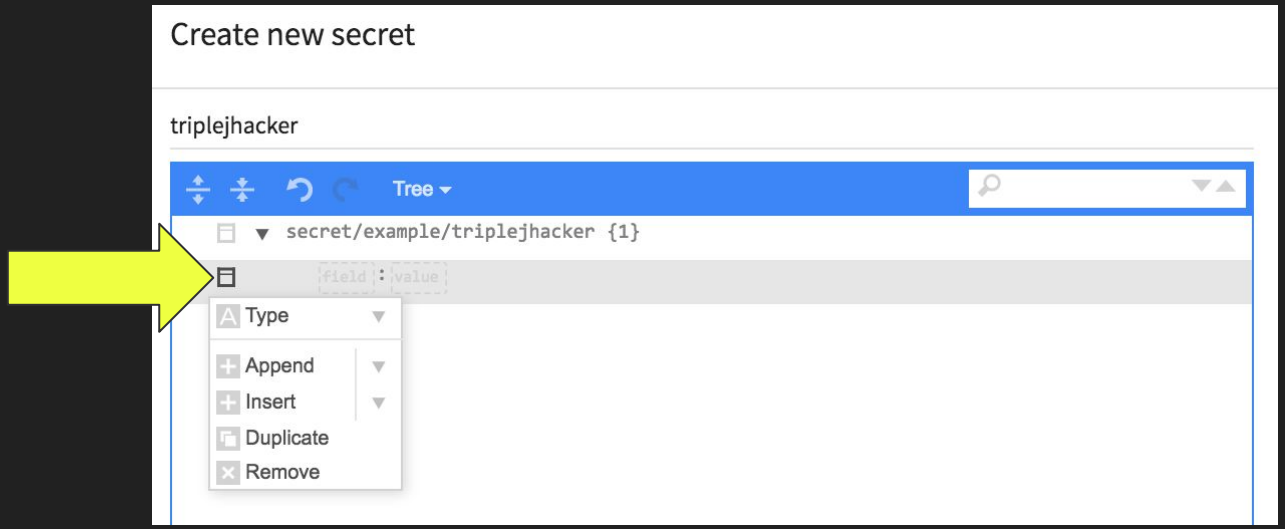
Access to Vault (GUI version)

- Click “test” item to view its key value
- Click “NEW SECRET” at the far right to create a new item



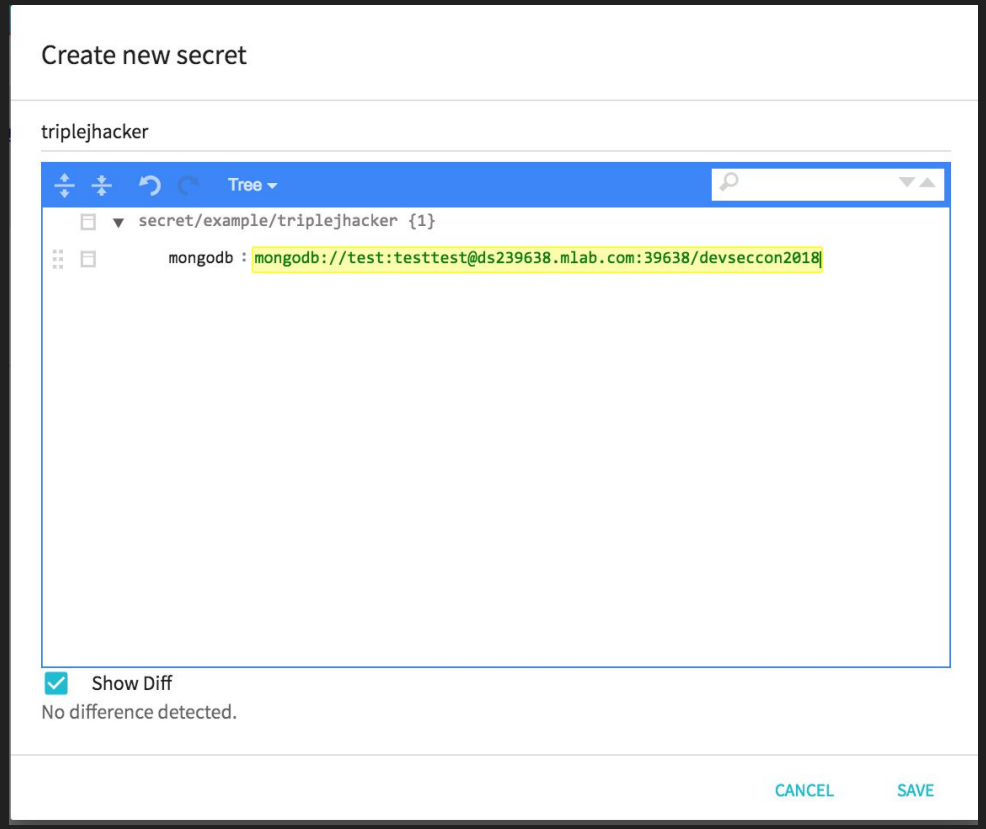
Access to Vault (GUI version)

- Fill <Insert object key> as your username
- Click on the box icon (arrow pointing) and choose “Append”, “field:value” boxes will appear



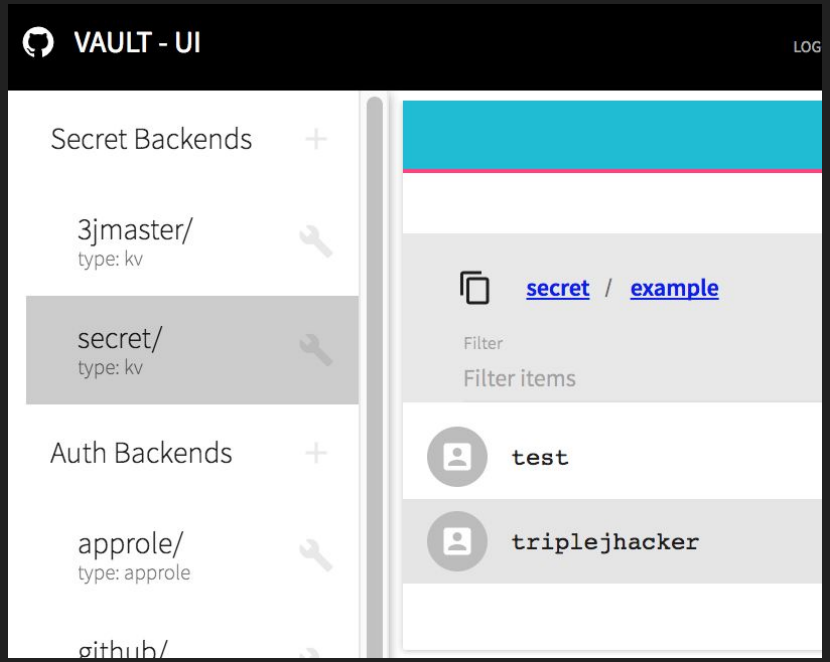
Access to Vault (GUI version)

- Fill “field” as ‘mongodb’
- Fill “value” as the secret mongodb URL in your app from slide 7
  - mongdb://...
- Click “Save”



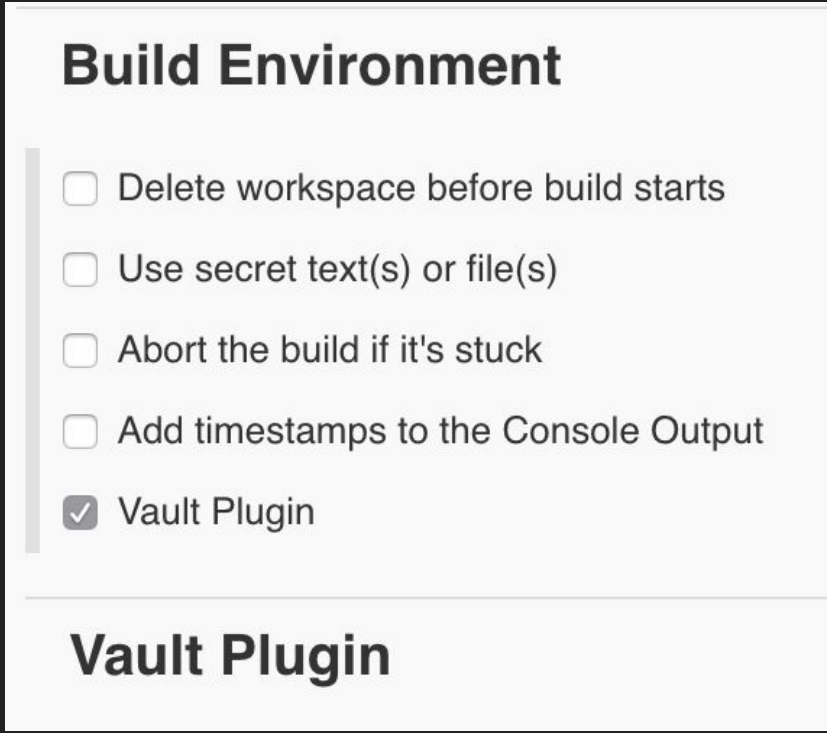
Access to Vault (GUI version)

- Your secret is created in Vault, click on it to view
- Note its path: i.e.secret/example/<username>



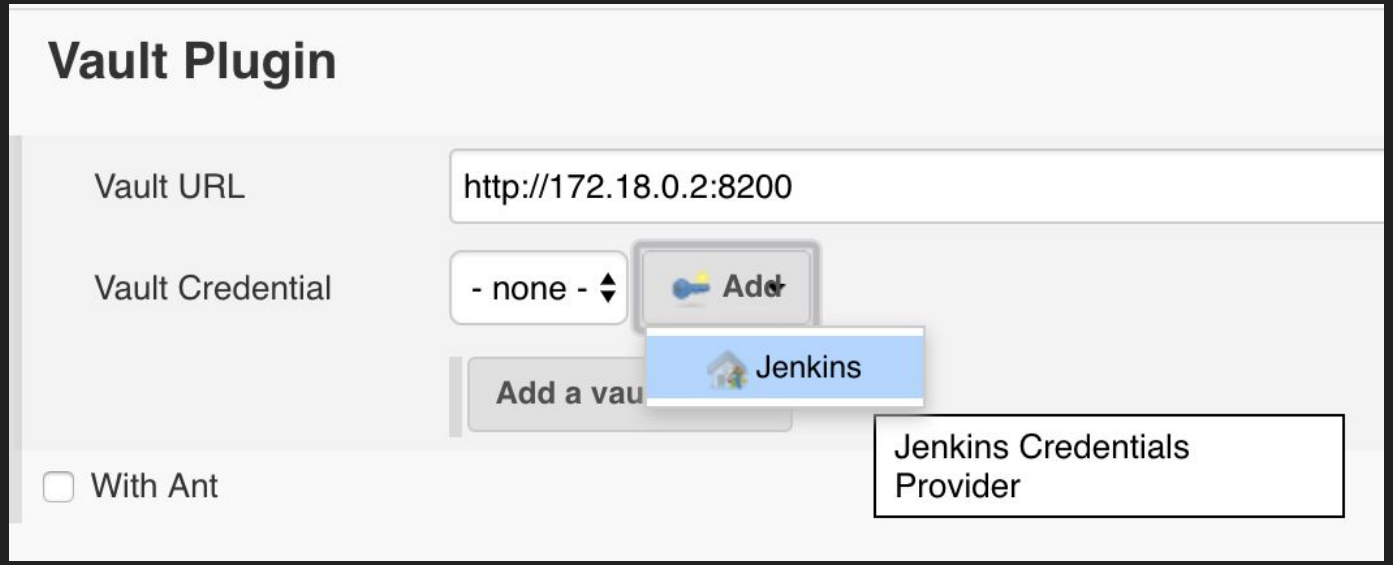
Retrieve secret from Vault with Jenkins

- Navigate back to your Jenkins project, under “Configure”
- Uncheck previous “Use secret text(s) file(s)” box
- Check “Vault Plugin” box



Retrieve secret from Vault with Jenkins

- Fill Vault URL as http://172.18.0.2:8200
- Click “Add” -> “Jenkins”



Retrieve secret from Vault with Jenkins (CLI version)

- In the same terminal, run:

```
$ vault token-lookup
Key                 Value
---                 -----
accessor            4fb12012-fb92-8d84-a4ed-bdb820532739
creation_time       1519349293
creation_ttl        2764800
display_name        github-triplejhacker
entity_id           ab3297fe-5fc2-5dca-f38e-c2716151774f
expire_time         2018-03-27T01:28:13.581448089Z
explicit_max_ttl    0
id                  ce38db15-****-524f-482c-************
issue_time          2018-02-23T01:28:13.581440761Z
meta                map[org:DevSecOpsSG username:triplejhacker]
num_uses            0
orphan              true
path                auth/github/login
policies            [default]
renewable           true
ttl                 2764132
```

This is your vault token

Retrieve secret from Vault with Jenkins (CLI version)

- In the same terminal, run:

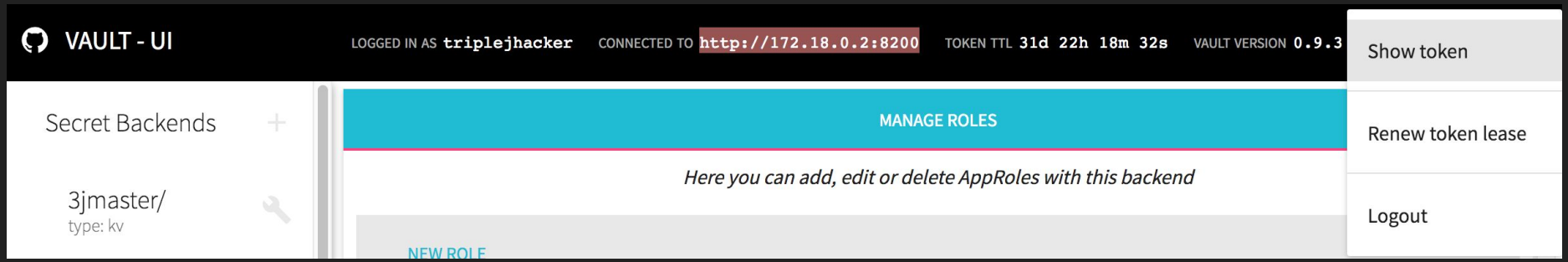
```
$ vault read auth/approle/role/example/role-id
Key       Value
---       -----
role_id   e4964208-6fed-882b-7739-ace170ec5aba
$ vault write -f auth/approle/role/example/secret-id
Key                 Value
---                 -----
secret_id           acb24ed4-1232-298a-abd4-4ad0ac77c461
secret_id_accessor  b84b516e-eb53-c5ff-8b5c-************
```

Secret-id is uniquely generated each time this command is ran

You can now skip slides for Retrieve secret from Vault with Jenkins (GUI version)

Retrieve secret from Vault with Jenkins (GUI version)

- Go to http://13.228.110.97:8300
- Click on the top right corner and “Show token”
- Copy the vault token
- *Note: This is generated by Vault and is different from the GitHub Token*



Retrieve secret from Vault with Jenkins (GUI version)

- Sorry, there's no GUI for this!
- Paste your vault token from the previous slide
- Using curl, or https://www.getpostman.com/apps or any request tool, to GET request:

```
curl --header "X-Vault-Token: <REPLACE WITH VAULT TOKEN>"
http://13.228.110.97:8200/v1/auth/approle/role/example/role-id
```

- Copy the **role-id**
  - {"request_id":"8d789757-ab4a-de80-9783-0927ac926f35","lease_id":"","renewable":false,"lease_duration":0,"data":{"role_id":"e4964208-6fed-882b-7739-ace170ec5aba"},"wrap_info":null,"warnings":null,"auth":null}

Retrieve secret from Vault with Jenkins (GUI version)

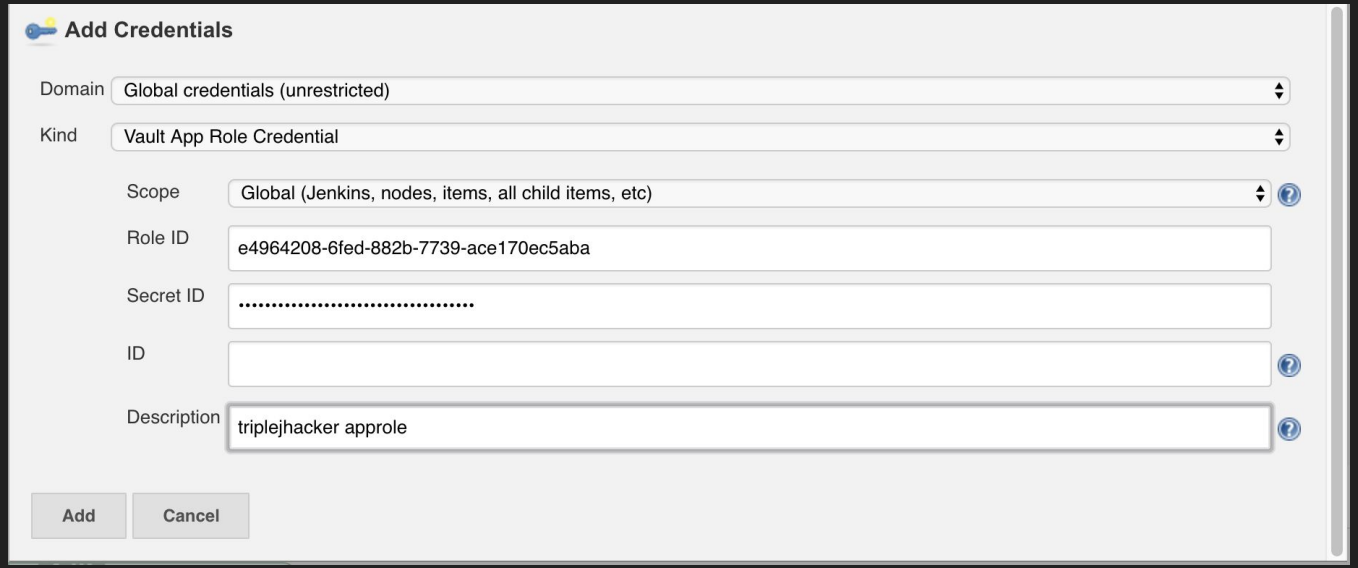
- Sorry, there's no GUI for this!
- Paste your vault token from the previous slide
- Using curl, or https://www.getpostman.com/apps or any request tool, to POST request:

```
curl --header "X-Vault-Token: <REPLACE WITH VAULT TOKEN>" --request POST
http://13.228.110.97:8200/v1/auth/approle/role/example/secret-id
```

- Copy the **secret-id**
  - {"request_id":"a2bdb1d5-28d9-d7c8-da4e-94800ba496e3","lease_id":"","renewable":false,"lease_duration":0,"data":{"secret_id":"a6f53a92-96bd-9fc4-9d8a-**********","secret_id_accessor":"60e54752-c6d1-320f-574a-a1ee3f7a219b"},"wrap_info":null,"warnings":null,"auth":null}

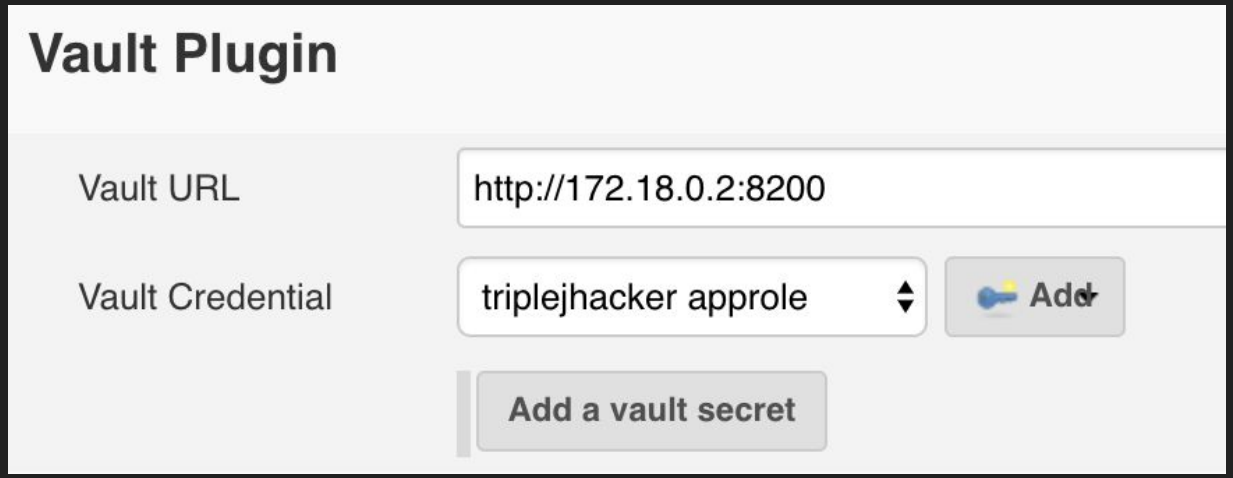
Store secrets within Jenkins

- Choose “Vault App Role Credential” as kind
- Fill Role ID from previous, previous slide
- Fill Secret ID from previous slide
- Fill Description with your username (for easy identification)
- Click “Add”



Store secrets within Jenkins

- Choose your newly created item as the Vault Credential i.e. “triplejhacker approle”
- Click “Add a vault secret”



Store secrets within Jenkins

- Fill Environment Variable as “mongodb” (all small caps)
- Fill the rest as illustrated matching from the Vault UI
- Click “Save”

Path
(secret/example/triplejhacker)

Editing se et/example/triplejhacker

Tree

secret/example/triplejhacker {1}

mongodb : mongodb://test:testtest@c

Key Name
(mongodb)

Vault Secret

Path secret/example/triplejhacker

Environment Variable mongodb

The environment variable to set with the value of the vault key.
(from HashiCorp Vault Plugin)

Key name mongodb

The vault key whose value with populate the environment variable.
(from HashiCorp Vault Plugin)

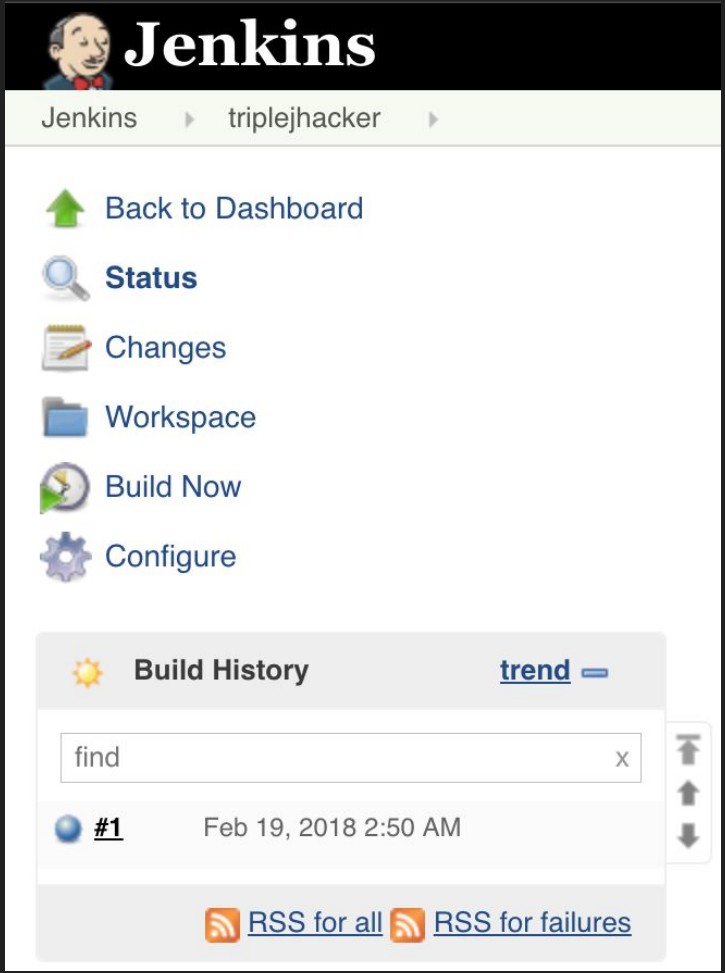
Add a key/value pair

Save Apply

Add a vault secret

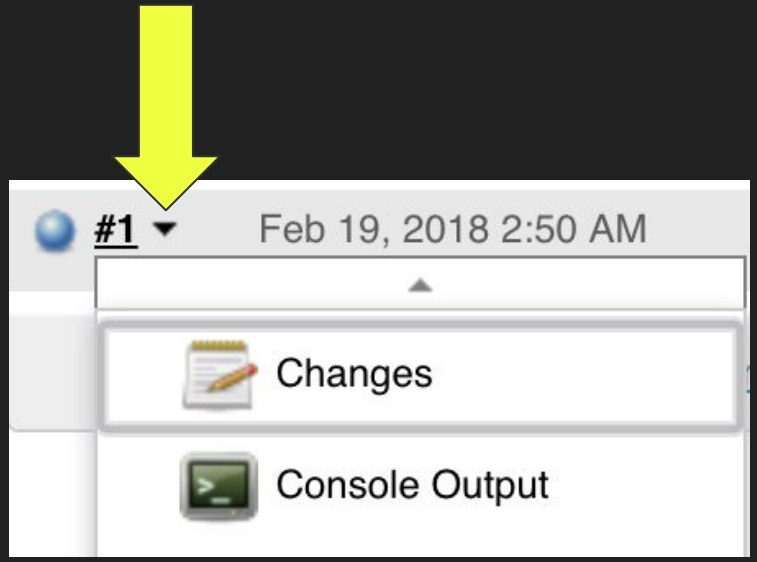
Build and Run with Jenkins

- Click “Build Now”
- Under “Build History”, there should be a build number like “#1” or “#2” or…



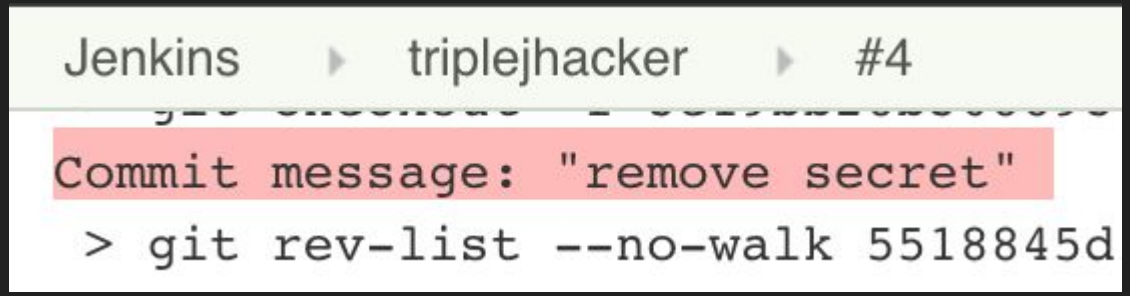
Build and Run with Jenkins

- Click on the arrow beside the build number (shown here)
- Click on “Console Output” to show the logs from the build



Build and Run with Jenkins

- If all goes well, it should be pulling from your latest code commit, check the commit message.
  - It should be the same as the value in “`git commit -m ‘remove secret’` ” ran earlier
- We didn’t push any code changes so this is correct.



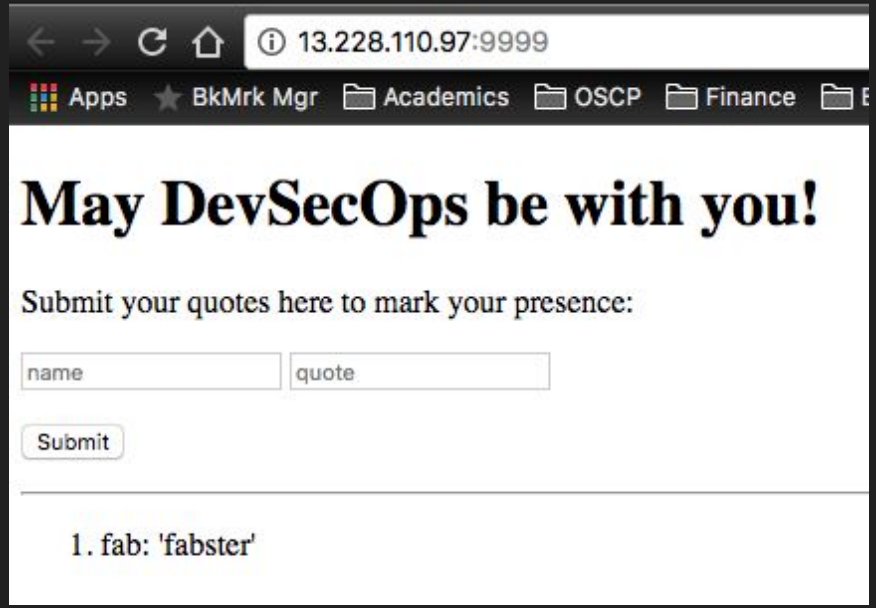
Build and Run with Jenkins

- If all goes well, it should look something like this ending with “**Finished: SUCCESS**”
- Note that the secret is also masked out. Good job Jenkins!

```
+ sudo docker run --network=isolated_nw -d -p 9999:3000 --name triplejhacker --env MONGODB_CREDENTIALS=****
triplejhacker/devseccon2018
2704aa320979f1c615df5db102fac4e0242bc5c6fbabada54c3809bbc4ba08a4
Finished: SUCCESS
```

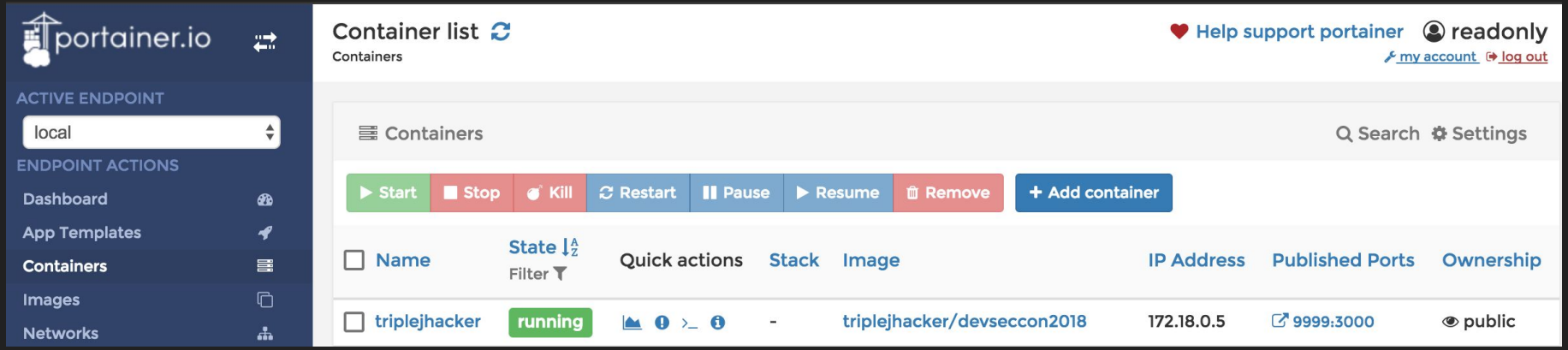
Access your deployed app

- Append the port number you specified in jenkins to http://13.228.110.97:<specified_port> and go to this URL in your browser
- In my example, I go to http://13.228.110.97:9999
- A simple app should still display
- Interact with the app by adding quotes
- These quotes are stored in your mongodb in mlab. You can go back to mlab and check the changes in database



Debug your deployed app

- Docker UI is at http://13.228.110.97:8100
- Login with username and password “readonly” to view the state and logs of containers



Congrats!
You have just removed a shameful secret AND use App Role to control access to the secrets!

But is this good enough?

https://www.vaultproject.io/

Solution 3 - Remove Secrets from App and Jenkins, using App Role and +++

… until next time...

https://www.vaultproject.io/docs/concepts/response-wrapping.html

Or try

https://medium.com/what-about-security/all-day-devops-2017-removing-developers-shameful-secrets-f5aca3960316

Thank you for your attention, patience, and enthusiasm during the workshop!

Happy Lunar New Year!
Cheers!