Aws – ssh – access remotely

Provisioning aws instance with

- Ansible Playbook
- Boto

What is boto - You use the AWS SDK for Python (Boto3) to create, configure, and manage AWS services, such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). The SDK provides an object-oriented API as well as low-level access to AWS services.

Install Boto3

Install the latest Boto3 release via **pip**:

pip install boto3

If your project requires a specific version of Boto3, or has compatibility concerns with certain versions, you may provide constraints when installing:

Install Boto3 version 1.0 specifically

pip install boto3==1.0.0

Make sure Boto3 is no older than version 1.15.0

pip install boto3>=1.15.0

Avoid versions of Boto3 newer than version 1.15.3

pip install boto3<=1.15.3

Sources: https://www.golinuxcloud.com/provision-aws-using-ansible/

Source: https://linuxhint.com/install_aws_cli_ubuntu/

Additional information needed for deployment manifest or service manifest

Configmap

Source: https://kubernetes.io/docs/concepts/configuration/configmap/

```
                    Application Config variables.
**** Dev

DBA_URL=https://database.com/utrain/dev
PORT_NUMBER=80
DBA_USER=dba-dev
DBA_PASSWORD=DevPassword
MONITOR_URL=https://prometheus.com/dev
MONITOR_USER=dev-monir
MONITOR_PASSWORD=DevMonPwd
WEB_CONTENT=/opt/dev/content
WEB_VOLUME=dev-volumes
```


```
                    Application Config variables.
**** Dev

DBA_URL=https://database.com/utrain/dev
PORT_NUMBER=80
DBA_USER=dba-dev
DBA_PASSWORD=DevPassword
MONITOR_URL=https://prometheus.com/dev
MONITOR_USER=dev-monir
MONITOR_PASSWORD=DevMonPwd
WEB_CONTENT=/opt/dev/content
WEB_VOLUME=dev-volumes
```

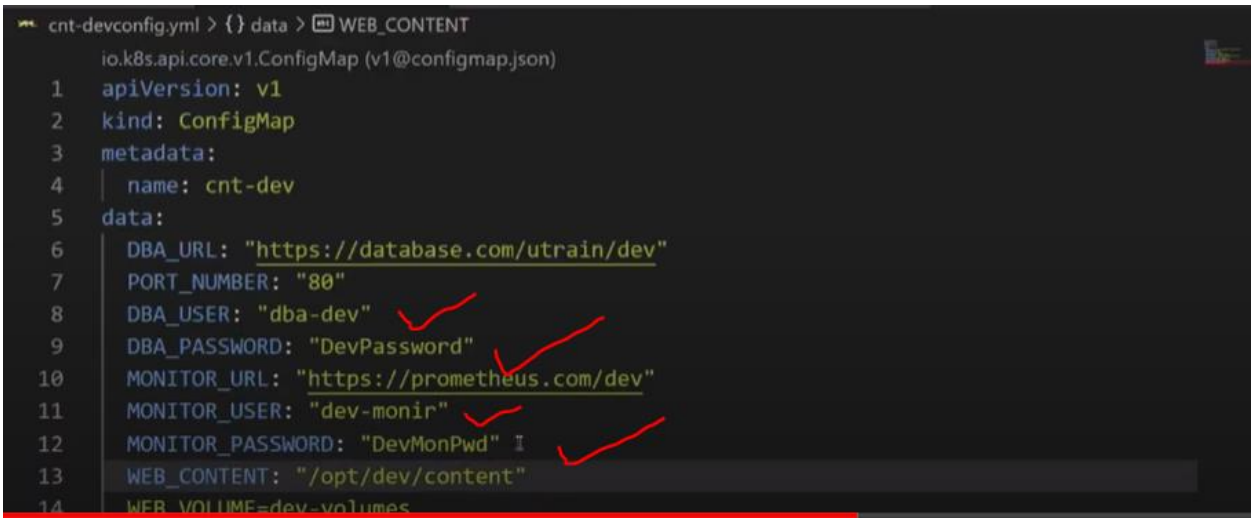

cnt-devconfig.yml > {} data > WEB_CONTENT

io.k8s.api.core.v1.ConfigMap (v1@configmap.json)

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cnt-dev
data:
  DBA_URL: "https://database.com/utrain/dev"
  PORT_NUMBER: "80"
  DBA_USER: "dba-dev"
  DBA_PASSWORD: "DevPassword"
  MONITOR_URL: "https://prometheus.com/dev"
  MONITOR_USER: "dev-monir"
  MONITOR_PASSWORD: "DevMonPwd"
  WEB_CONTENT: "/opt/dev/content"
  WEB_VOLUME=dev-volumes
```

All the username and the password for the application will be stored in the

Secret.yaml file



How to encrypt your password for the variables using base64

```
Forex@LAPTOP-4C614INS MINGW64 ~/kube/cnt-app (master)
$ echo dba-dev |base64
ZGJhLWRldgo=

Forex@LAPTOP-4C614INS MINGW64 ~/kube/cnt-app (master)
$ echo Devpassword |base64
RGV2cGFzc3dvcmQK

Forex@LAPTOP-4C614INS MINGW64 ~/kube/cnt-app (master)
$
```

You can also Decode

```
Forex@LAPTOP-4C614INS MINGW64 ~/kube/cnt-app (master)
$ echo RGV2cGFzc3dvcmQK |base64 --decode
Devpassword
```

Prometheus and Grafana – If application are broken inside the pod, Kubernetes does not have a way to monitor this.

You need to monitor what is happening inside the pod and the only way to do it is to do is by the use of monitoring tool that are capable of monitoring the modern application.

Let say you want to set up the Prometheus and Grafana

You will need the following

- Service for both Grafana and Prometheus
- Configmap for the variables for both
- Secret for the username and password for both

This is a lot of work

In a company, you are going to either use HELM or KUBECTL to deploy your manifests

Let Deploy

kubectl create namespace prod

```
  556  kubectl create namespace qa
  557  kubectl create namespace dev
  558  clear
  559  ls
  560  kubectl apply -f cnt-dev-secret.yaml
  561  kubectl apply -f cnt-dev-secret.yaml
  562  kubectl apply -f cnt-qa-secret.yaml
  563  kubectl apply -f cnt-prod-scret.yaml
  564  kubectl apply -f cnt-qa-secret.yaml
  565  kubectl get secret
```

kubectl get secret -n dev

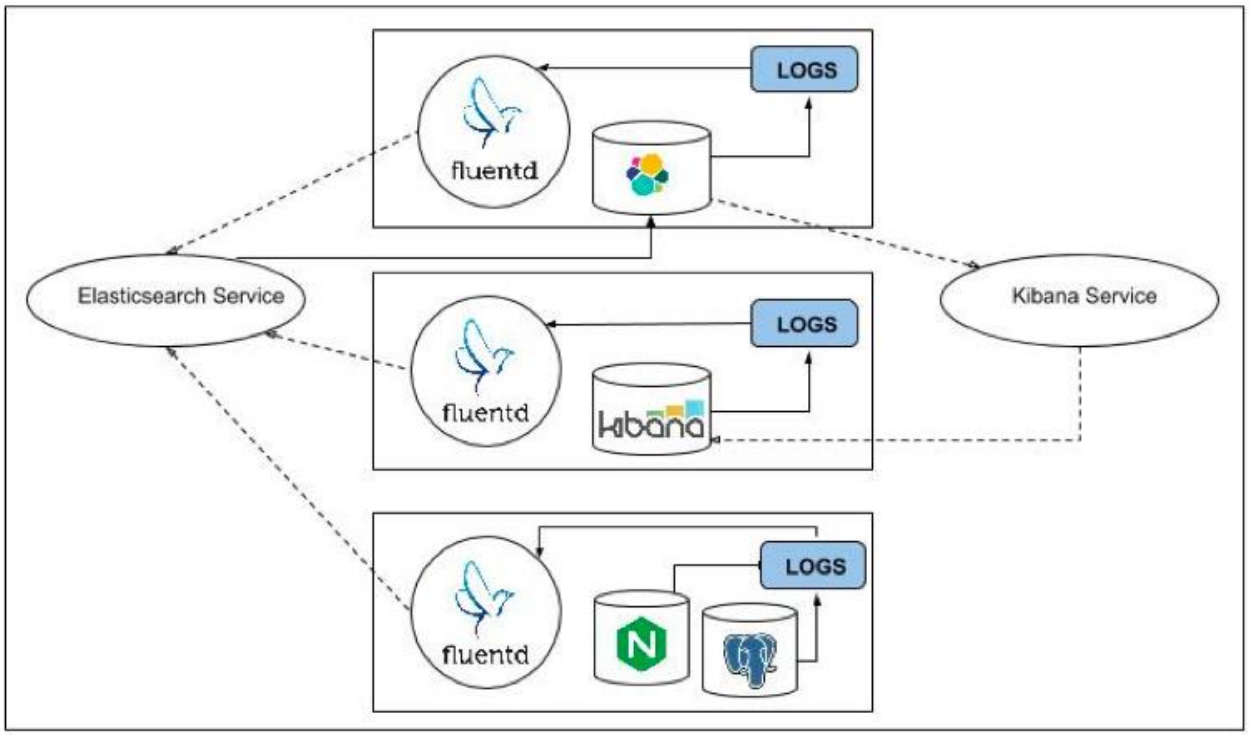
```
  571  kubectl get secret -n qa
```

572 kubectl get secret -n prod
kubectl describe secret cnt-prod-secret -n prod

Create an Alias

alias get= ‘kubectl get’

ELK = elastic search + fluentd + Kibana



Analyzing kubernetes logs in kibana
Now that we have our logs stored in elasticsearch, the next step is to display them in kibana. to do this, we will need to run kibana in our cluster. just as with elasticsearch, we need only one kibana instance.

Source: https://dzone.com/articles/kubernetes-log-analysis-with-fluentd-elasticsearch

HELM

Choco Install Kubernetes-helm

Helm version

Resource Helm.sh

This is a website where everything has been set for you

Deployment manifest / service manifest /secret /configmap