**Demo1:** 1-cfn-s3bucket.yaml

( Create a Simple s3 Bucket)

* Go to CFN Template reference :
* <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html>
* Use cfn-s3bucket.yaml as the first demo
* From commandline
  + aws cloudformation create-stack --stack-name mydemostack --template-body <file://cfn-s3bucket.yaml>

Cleanup

* aws s3api delete-bucket --bucket ik-cfndemo-bucket1

**Demo2 :** 2-ec2.yaml

EC2 with Userdata

**Demo3:** 3-ec2-cfninit.yaml

(Explain the following)

* Userdata is procedural
* Not easy to manage
* OS and distribution specific (eg use yum or apt-get)
* Cloudformationint passes directives to cfninit process
  + Attempts to be os independent
  + Desired state engine not a procedural one
  + Supports config sets ( can use different config sets)
    - Default is config if it is not specified. We can have multiple configsets then we can explicitly call the configset. For eg we can have one config set to activate something and one config set to deactivate something
  + Order is significant
  + Allows control of timing(in some cases)( for eg do a step only after reboot)
  + It allows authentication (from s3 resources for eg)
  + Files and folders can be created from online sources
  + It is idempotent. If it is done it will not redo it

( it is specified with metadata key)

( config set has groupings and the order of these groupings is important

* Config:

**Packages** – to install packages if they are not installed. If version not specified eg apache and you have not specified the version then it will not renstall the latest version that you have desired

**Groups** – linux groups

**Users** – users created do not have shell logins by default meant for programs

**Sources** – files, tar, zip files from reomote sources to be downloaded and unzipped locally. Authentication is supported

**Files** – Works on single files. Move files from remote locations to local files. Create file using content directive. Eg database config file in php program

**Commands –** To specify commands to be run in order specified

**Services** – See the eg below

services:

sysvinit:

ngnix:

enabled: true

ensureRunning : true

files:

-“/etc/ngnix/ngnix.conf”

sources:

“/var/www/html”

To call cfninit add this line to userdata section

/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource EC2server --configsets config --region ${AWS::Region}

Cfnhup

* Listens to cfninit metadata changes and based on the changes can be made to run arbitrary commands
* The configuration file is store in /etc/cfn/cfn-hup.conf (Global conf)
* Second location is /etc/cfn/hooks.d/,,,conf – this contains hooks. Hooks are triggers when to run and what to run
* Default it checks every 15mins to see if any metadata has changed

**Demo Procedure**

EC2 with cfninit process

Run the cloudformation template

Modify the file contents and run update. We see that the file does not get updated as cfninit will only ensure the file exists. It will Not modify it

**Demo 4:** 4-ec2-cfnhup.yaml

Expalin the YAML files new content for running cfn-hup process

* Cfn-hup.conf
* Cfn auto reloader
* Cfn-hup process

Now modify the file contents of html file

Update the stack

Open the /var/log/cfn-hup.log

After some time about 5 mins it will reload and you can see the modifications in the html file