DEPARTMENT OF COMPUTER ENGINEERING & APPLICATIONS
INSTITUTE OF ENGINEERING & TECHNOLOGY

B. Tech. IV Year

Project Report

On

"DevOps"

Submitted by
1. Ram kamra (G-52/ 161500437)
2. Hitesh Thandel (E-23/ 161500242)

Odd Semester, 2019-20

Index

Chapter 1
Introduction

1.1 Motivation and Overview

Cloud computing is a term referred to storing and accessing data over the internet. It doesn't store any data on the hard disk of your personal computer. In cloud computing, you can access data from a remote server. Cloud computing has disrupted the various methodologies of software development and hence DevOps is a technology to cope up with the changing needs of software development.

1.2 Objective

DevOps is a set of practices that combines software development and information technology operations which aims to shorten the systems development life cycle and provide continuous delivery with high software quality.
Our objective here is to learn about the various tools used in the methodology called as DevOps. It is very necessary to evolve with emerging trend and hence learning DevOps tools like AWS, Jenkins , Git, etc is necessary.These tools are also often new technologies that have only become available in the last 4-5 years around the time that DevOps was born.

1.3 Scope

In future due to the advent of cloud technologies the management, support and production of a software will completely change and hence DevOps acts as a methodolody to bring this change.. Moreover, this shift in the methodologies have broiught more agility to the the entire process and has not only made the manual task automated but also has made the process more efficient, thus resulting in better quality, availability and scalability of the product.

Chapter 2

Software Requirement Analysis

2.1 Introduction

2.1.1 Objectives

To understand the architecture of AWS cloud platform and understand the basics of different DevOps tools like Jenkins, Git, Ansible.

2.1.2 Scope

In future due to the advent of cloud technologies the management, support and production of a software will completely change and hence DevOps acts as a methodology to bring this change. Moreover, this shift in the methodologies have brought more agility to the the entire process and has not only made the manual task automated but also has made the process more efficient, thus resulting in better quality, availability and scalability of the product.

2.1.3 Overview

DevOps is a set of practices that combines software development and information technology operations which aims to shorten the systems development life cycle and provide continuous delivery with high software quality. We have had various tasks during our training.

Our goal here was to learn about the various tools used in the methodology called DevOps. It is very necessary to evolve with emerging trend and hence learning DevOps tools like AWS, Jenkins , Git, etc is necessary.These tools are also often new technologies that have only become available in the last 4-5 years around the time that DevOps was born.

2.1.4 References

- https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

2.2 General Description

2.2.1 General constraints

- Personal computer with minimum 8 GB RAM
- High Speed Internet Connection

2.2.2 Assumptions & Dependencies

- Completion time is 1 months
- Each team member should work at least for 5 hours a day.

2.3 Specific Requirements

2.3.1 Hardware Interface

- Personal computer with minimum 8 GB RAM

2.3.2 Software Interface

- AWS
- Jenkins
- Git

2.4 Softwares in Detail

2.4.1 Jenkins

Jenkins offers a simple way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks. While Jenkins doesn't eliminate the need to create scripts for individual steps, it does give you a faster and more robust way to integrate your entire chain of build, test, and deployment tools than you can easily build yourself.

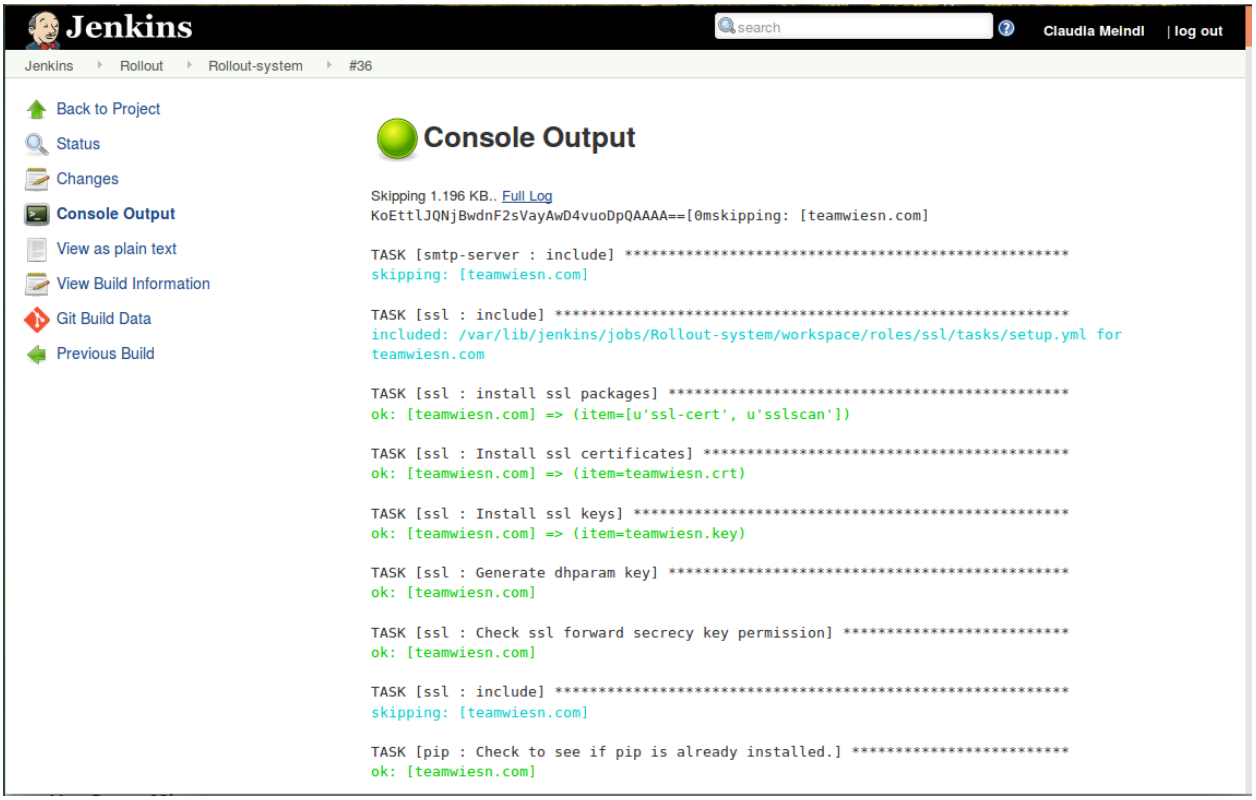


Fig 1. Jenkins

2.4.2 GIT

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano. As with most other distributed version-control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server. Git is free and open-source software distributed under the terms of the GNU General Public License version 2.

2.4.4 AWS

Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis. In aggregate, these cloud computing web services provide a set of primitive abstract technical infrastructure and distributed computing building blocks and tools. One of these services is Amazon Elastic Compute Cloud, which allows users to have at their disposal a virtual cluster of computers, available all the time, through the Internet.

Chapter 3
AWS

What is Cloud Computing?

Cloud computing is a term referred to storing and accessing data over the internet. It doesn't store any data on the hard disk of your personal computer. In cloud computing, you can access data from a remote server.

What is AWS?

Amazon web service is a platform that offers flexible, reliable, scalable, easy-to-use and cost-effective cloud computing solutions.

AWS is a comprehensive, easy to use computing platform offered Amazon. The platform is developed with a combination of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.

History of AWS

- 2002- AWS services launched
- 2006- Launched its cloud products
- 2012- Holds first customer event
- 2015- Reveals revenues achieved of $4.6 billion
- 2016- Surpassed $10 billon revenue target
- 2016- Release snowball and snowmobile
- 2019- Offers nearly 100 cloud services

Important AWS Services

Fig 2 Important AWS Services

AWS Compute Services

Here, are Cloud Compute Services offered by Amazon:

1. EC2(Elastic Compute Cloud) - EC2 is a virtual machine in the cloud on which you have OS level control. You can run this cloud server whenever you want.
2. LightSail-This cloud computing tool automatically deploys and manages the computer, storage, and networking capabilities required to run your applications.
3. Elastic Beanstalk— The tool offers automated deployment and provisioning of resources like a highly scalable production website.
4. EKS (Elastic Container Service for Kubernetes)—The tool allows you to Kubernetes on Amazon cloud environment without installation.
5. AWS Lambda—This AWS service allows you to run functions in the cloud. The tool is a big cost saver for you as you to pay only when your functions execute.

Migration

Migration services used to transfer data physically between your datacenter and AWS.

1. Snowball—Snowball is a small application which allows you to transfer terabytes of data inside and outside of AWS environment.

Storage

1. Amazon Glacier- It is an extremely low-cost storage service. It offers secure and fast storage for data archiving and backup.
2. Amazon Elastic Block Store (EBS)- It provides block-level storage to use with Amazon EC2 instances. Amazon Elastic Block Store volumes are network-attached and remain independent from the life of an instance.
3. AWS Storage Gateway- This AWS service is connecting on-premises software applications with cloud-based storage. It offers secure integration between the company's on-premises and AWS's storage infrastructure.

Security Services

1. IAM (Identity and Access Management)— IAM is a secure cloud security service which helps you to manage users, assign policies, form groups to manage multiple users.
2. Inspector—It is an agent that you can install on your virtual machines, which reports any security vulnerabilities.
3. Certificate Manager—The service offers free SSL certificates for your domains that are managed by Route53.
4. WAF (Web Application Firewall)— WAF security service offers application-level protection and allows you to block SQL injection and helps you to block cross-site scripting attacks.
5. Cloud Directory—This service allows you to create flexible, cloud-native directories for managing hierarchies of data along multiple dimensions.
6. KMS (Key Management Service)—It is a managed service. This security service helps you to create and control the encryption keys which allows you to encrypt your data.
7. Organizations—You can create groups of AWS accounts using this service to manages security and automation settings.

8. Shield—Shield is managed DDoS (Distributed Denial of Service protection service). It offers safeguards against web applications running on AWS.
9. GuardDuty —It offers threat detection to protect your AWS accounts and workloads.

Database Services

1. Amazon RDS- This Database AWS service is easy to set up, operate, and scale a relational database in the cloud.
2. Amazon DynamoDB- It is a fast, fully managed NoSQL database service. It is a simple service which allow cost-effective storage and retrieval of data. It also allows you to serve any level of request traffic.
3. Amazon ElastiCache- It is a web service which makes it easy to deploy, operate, and scale an in-memory cache in the cloud.
4. Neptune- It is a fast, reliable and scalable graph database service.
5. Amazon RedShift-It is Amazon's data warehousing solution which you can use to perform complex OLAP queries.

Application Services

1. Step Functions—It is a way of visualizing what's going inside your application and what different microservices it is using.
2. SWF (Simple Workflow Service)—The service helps you to coordinate both automated tasks and human-led tasks.
3. SNS (Simple Notification Service)—You can use this service to send you notifications in the form of email and SMS based on given AWS services.
4. SQS (Simple Queue Service)—Use this AWS service to decouple your applications. It is a pull-based service.
5. Elastic Transcoder—This AWS service tool helps you to changes a video's format and resolution to support various devices like tablets, smartphones, and laptops of different resolutions.

Deployment and Management

1. AWS CloudTrail**:** The services records AWS API calls and send backlog files to you.

2. Amazon CloudWatch**:** The tools monitor AWS resources like Amazon EC2 and Amazon RDS DB Instances. It also allows you to monitor custom metrics created by user's applications and services.
3. AWS CloudHSM**:** This AWS service helps you meet corporate, regulatory, and contractual, compliance requirements for maintaining data security by using the Hardware Security Module(HSM) appliances inside the AWS environment.

Developer Tools

1. CodeStar—Codestar is a cloud-based service for creating, managing, and working with various software development projects on AWS.
2. CodeCommit— It is AWS's version control service which allows you to store your code and other assets privately in the cloud.
3. CodeBuild—This Amazon developer service help you to automates the process of building and compiling your code.
4. CodeDeploy—It is a way of deploying your code in EC2 instances automatically.
5. CodePipeline—It helps you create a deployment pipeline like testing, building, testing, authentication, deployment on development and production environments.
6. Cloud9 —It is an Integrated Development Environment for writing, running, and debugging code in the cloud.

Mobile Services

1. Mobile Hub—Allows you to add, configure and design features for mobile apps.
2. Cognito—Allows users to signup using his or her social identity.
3. Device Farm—Device farm helps you to improve the quality of apps by quickly testing hundreds of mobile devices.
4. AWS AppSync —It is a fully managed GraphQL service that offers real-time data synchronization and offline programming features.

Business Productivity

1. Alexa for Business—It empowers your organization with voice, using Alexa. It will help you to Allows you to build custom voice skills for your organization.
2. Chime—Can be used for online meeting and video conferencing.
3. WorkDocs—Helps to store documents in the cloud

4. WorkMail—Allows you to send and receive business emails.

Desktop & App Streaming

1. WorkSpaces—Workspace is a VDI (Virtual Desktop Infrastructure). It allows you to use remote desktops in the cloud.
2. AppStream —A way of streaming desktop applications to your users in the web browser. For example, using MS Word in Google Chrome.

Artificial Intelligence

1. Lex—Lex tool helps you to build chatbots quickly.
2. Polly— It is AWS's text-to-speech service allows you to create audio versions of your notes.
3. Rekognition —It is AWS's face recognition service. This AWS service helps you to recognize faces and object in images and videos.
4. SageMaker—Sagemaker allows you to build, train, and deploy machine learning models at any scale.
5. Transcribe— It is AWS's speech-to-text service that offers high-quality and affordable transcriptions.
6. Translate—It is a very similar tool to Google Translate which allows you to translate text in one language to another.

AR & VR (Augmented Reality & Virtual Reality)

1. Sumerian—Sumerian is a set of tool for offering high-quality virtual reality (VR) experiences on the web. The service allows you to create interactive 3D scenes and publish it as a website for users to access.

Customer Engagement

1. Amazon Connect—Amazon Connect allows you to create your customer care center in the cloud.
2. Pinpoint—Pinpoint helps you to understand your users and engage with them.
3. SES (Simple Email Service)—Helps you to send bulk emails to your customers at a relatively cost-effective price.

Game Development

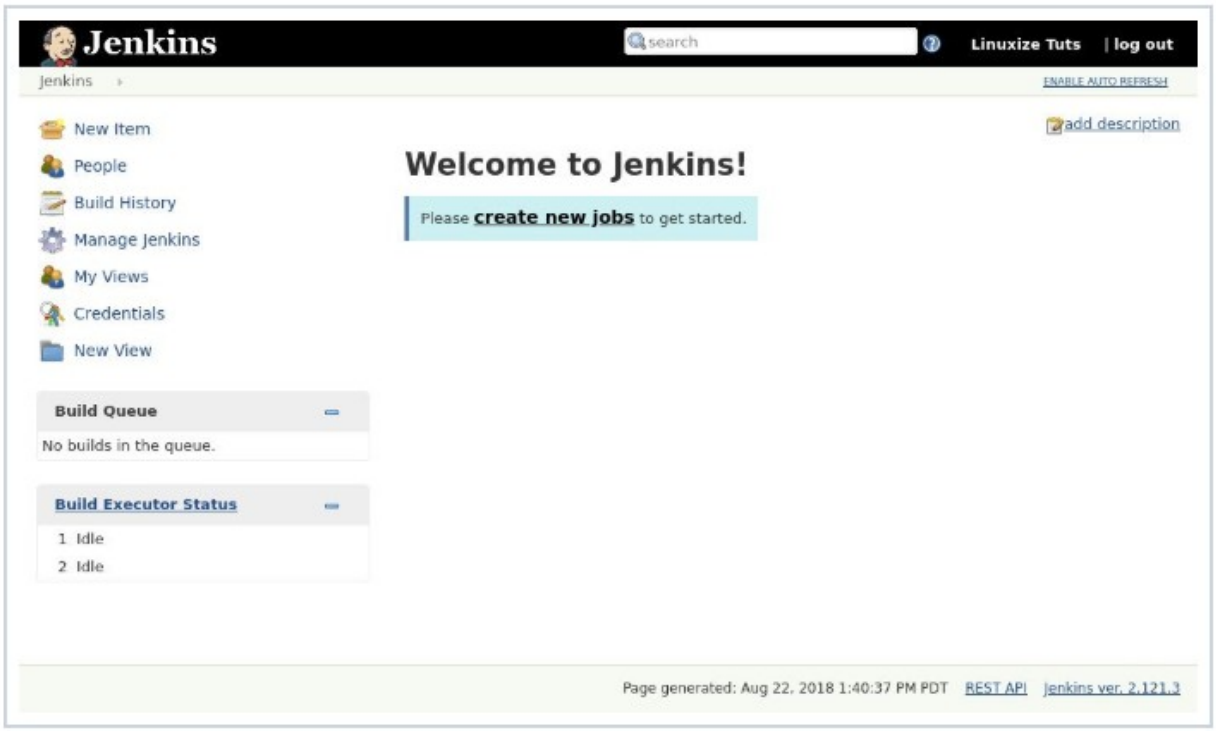
1. GameLift- It is a service which is managed by AWS. You can use this service to host dedicated game servers. It allows you to scale seamlessly without taking your game offline.

Advantages of AWS

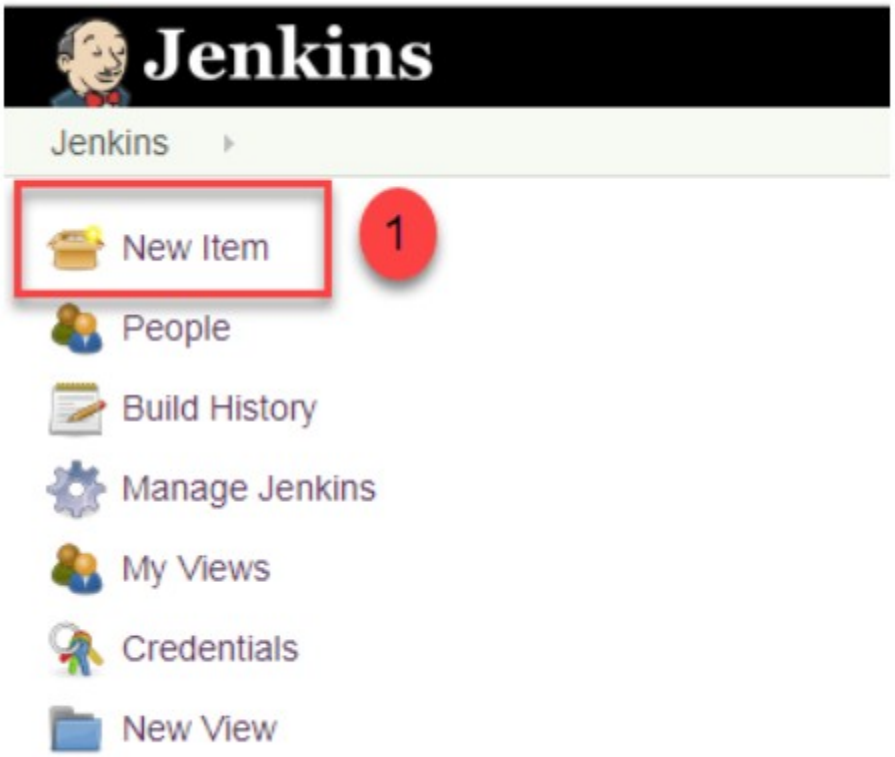
Following are the pros of using AWS services:

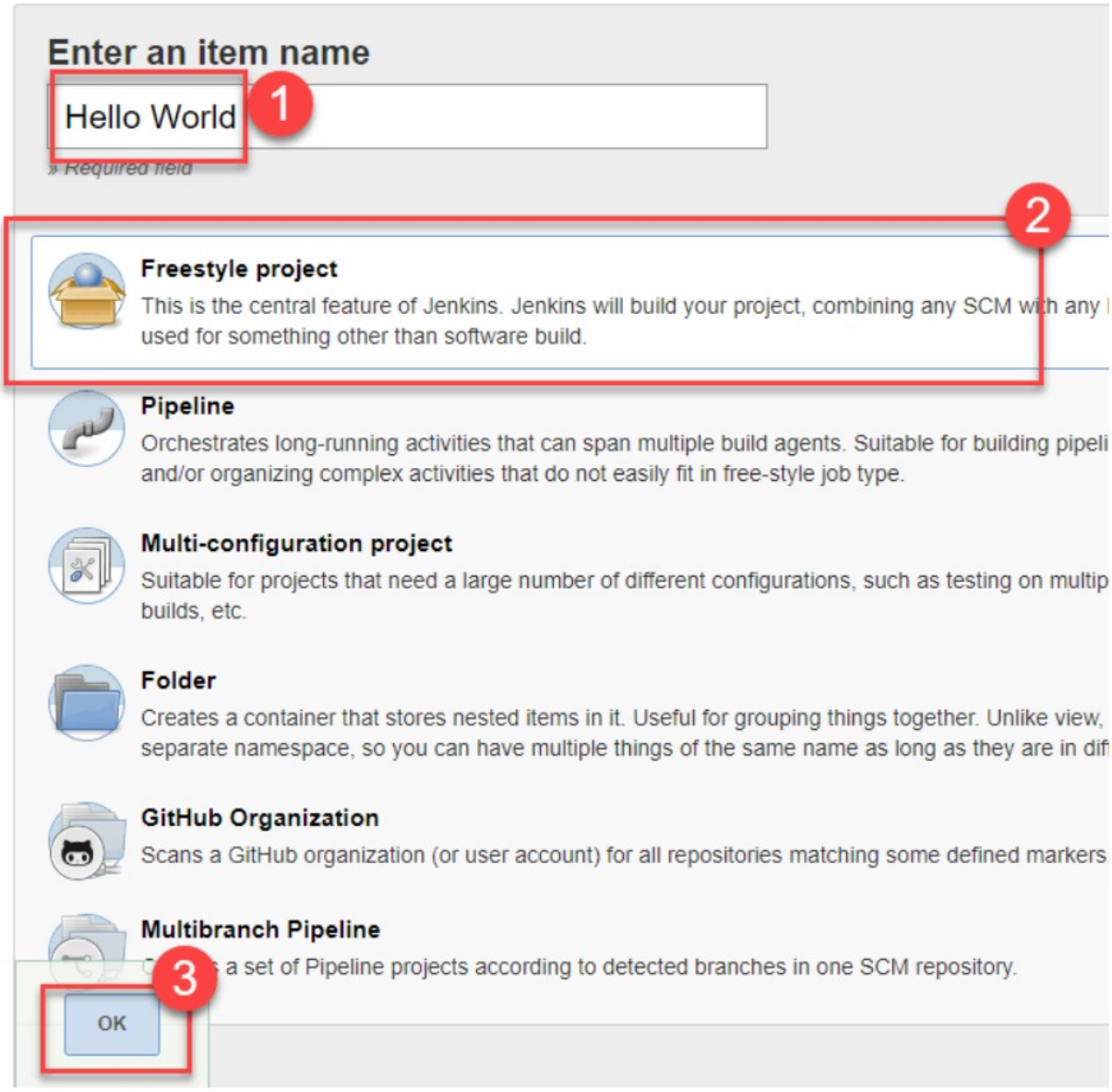
- AWS allows organizations to use the already familiar programming models, operating systems, databases, and architectures.
- It is a cost-effective service that allows you to pay only for what you use, without any up-front or long-term commitments.
- You will not require to spend money on running and maintaining data centers.
- Offers fast deployments
- You can easily add or remove capacity.
- You are allowed cloud access quickly with limitless capacity.
- Total Cost of Ownership is very low compared to any private/dedicated servers.
- Offers Centralized Billing and management
- Offers Hybrid Capabilities
- Allows you to deploy your application in multiple regions around the world with just a few clicks

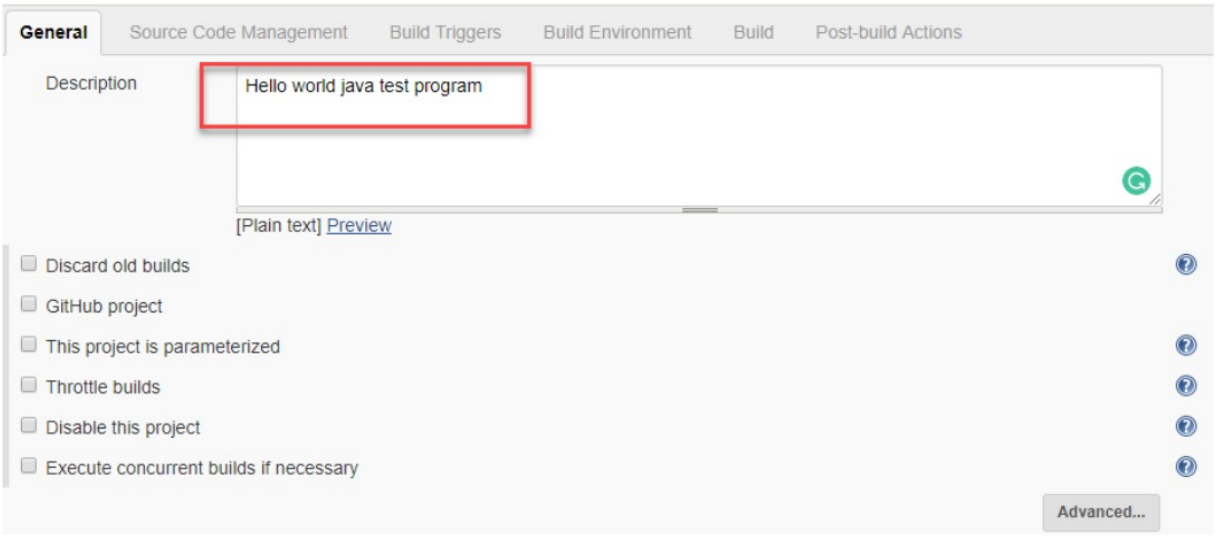
Chapter 4
Jenkins

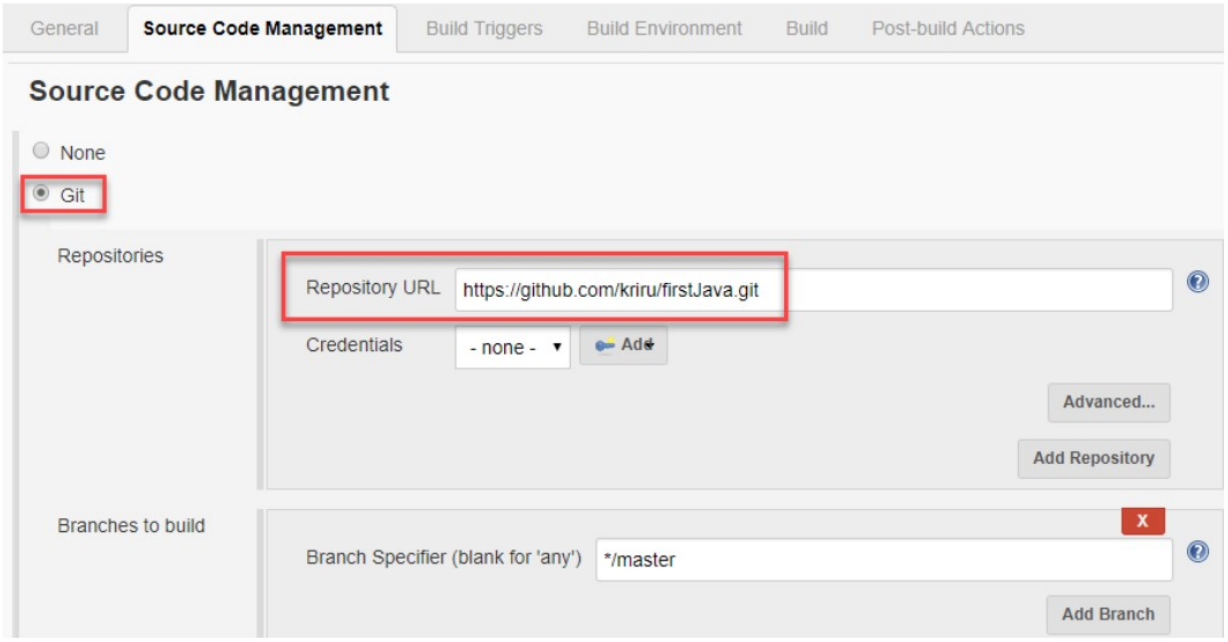


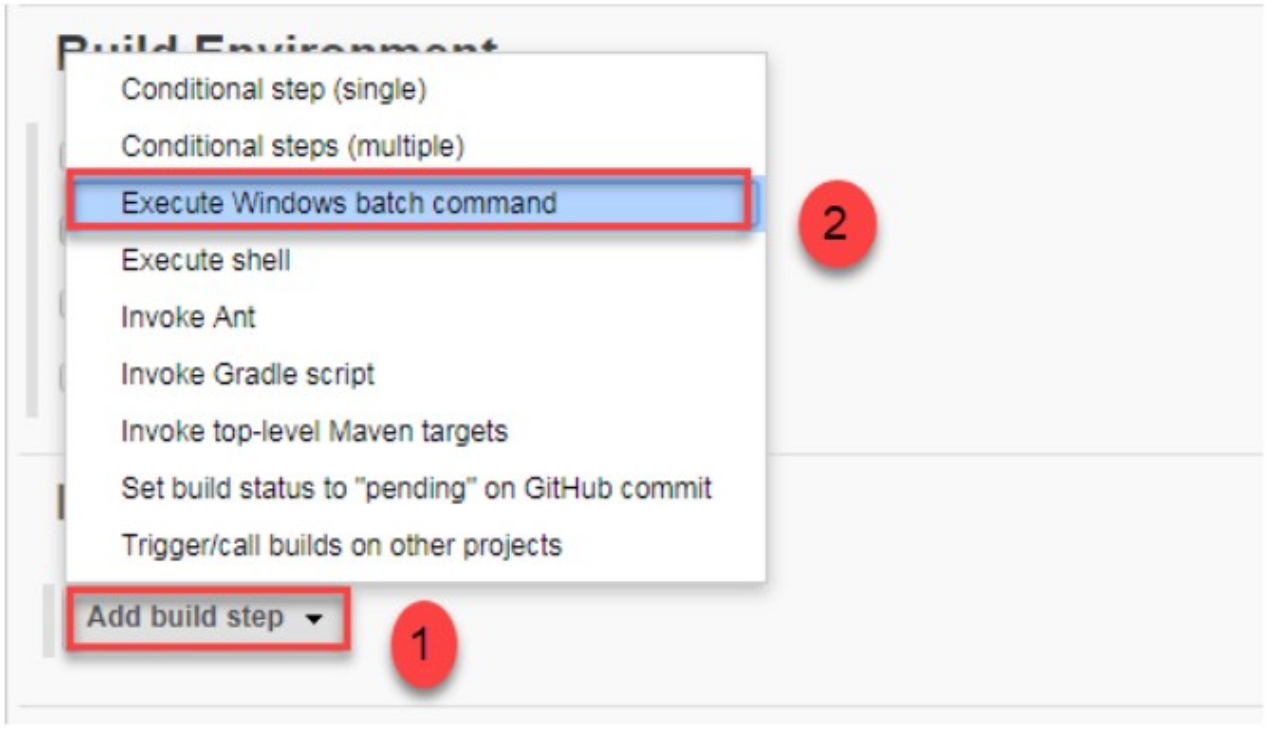
1. Create first job in Jenkins

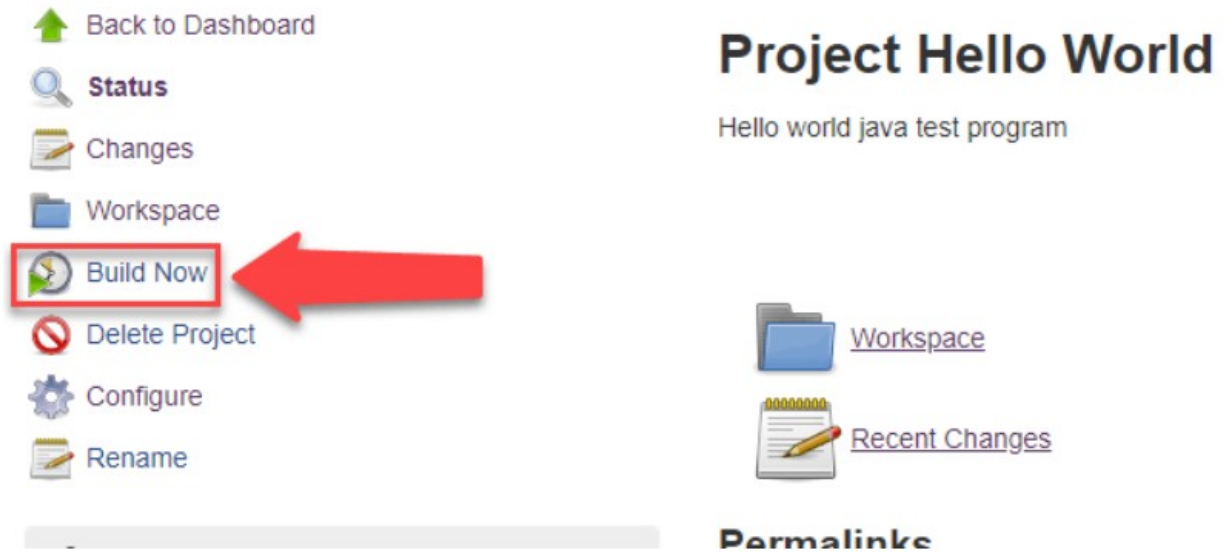

Jenkins
Jenkins
New Item
1
People
Build History
Manage Jenkins
My Views
Credentials
New View

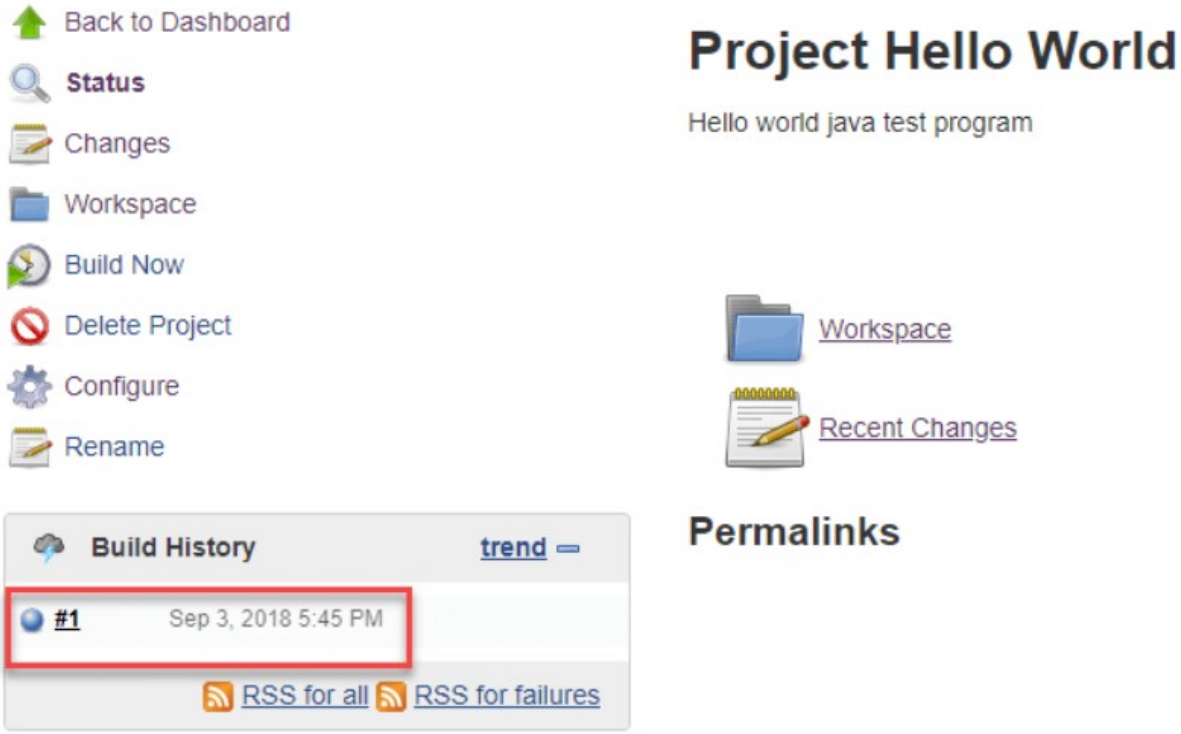

Enter an item name
Hello World
1
» Required field
2
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any
used for something other than software build.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipeli
and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multip
builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view,
separate namespace, so you can have multiple things of the same name as long as they are in dif
GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers
Multibranch Pipeline
a set of Pipeline projects according to detected branches in one SCM repository.
3
OK

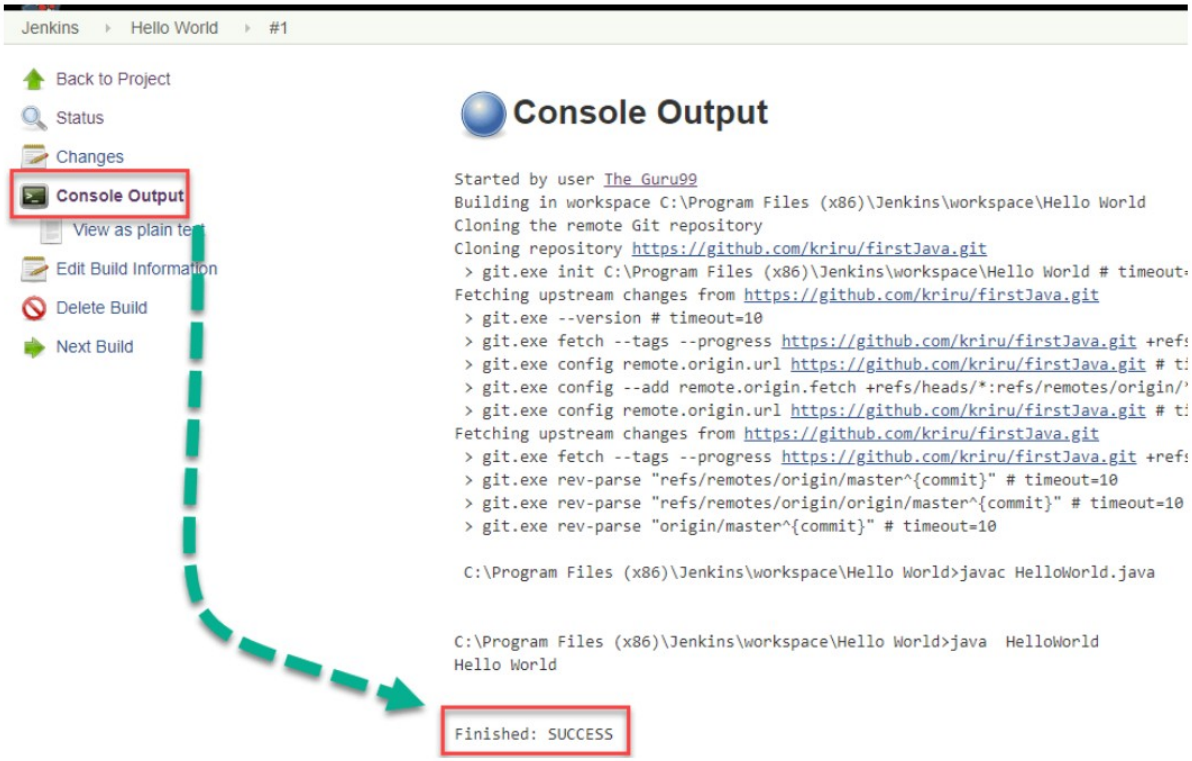

General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
Hello world java test program
[Plain text] Preview
Discard old builds
GitHub project
This project is parameterized
Throttle builds
Disable this project
Execute concurrent builds if necessary
Advanced...


General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/kriru/firstJava.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch


Conditional step (single)
Conditional steps (multiple)
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Set build status to "pending" on GitHub commit
Trigger/call builds on other projects
Add build step
1
2

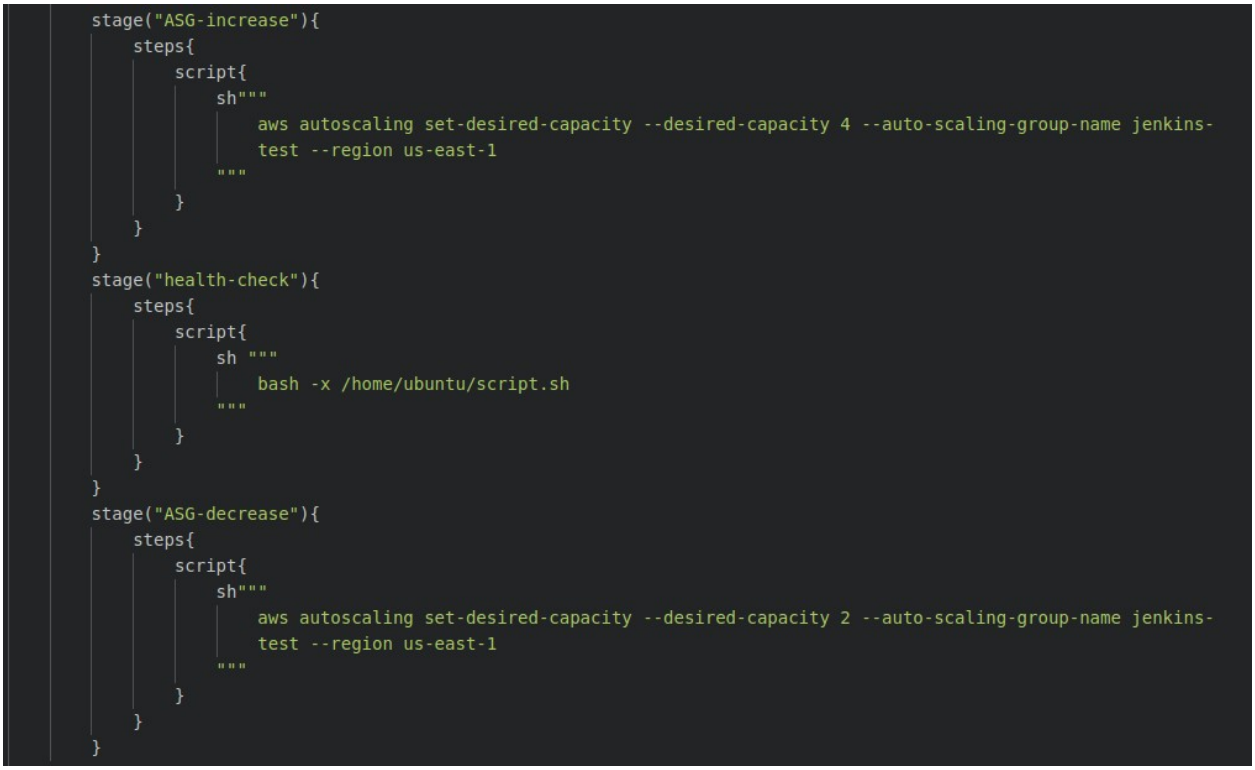

Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Rename
Project Hello World
Hello world java test program
Workspace
Recent Changes
Permalinks


Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Rename
Project Hello World
Hello world java test program
Workspace
Recent Changes
Permalinks
Build History
trend
#1
Sep 3, 2018 5:45 PM
RSS for all
RSS for failures



Jenkins Pipeline

```
stage("ASG-increase"){
    steps{
        script{
            sh"""
                aws autoscaling set-desired-capacity --desired-capacity 4 --auto-scaling-group-name jenkins-
                test --region us-east-1
            """
        }
    }
}
stage("health-check"){
    steps{
        script{
            sh """
                bash -x /home/ubuntu/script.sh
            """
        }
    }
}
stage("ASG-decrease"){
    steps{
        script{
            sh"""
                aws autoscaling set-desired-capacity --desired-capacity 2 --auto-scaling-group-name jenkins-
                test --region us-east-1
            """
        }
    }
}
```



```
stage("ASG-increase"){
    steps{
        script{
            sh"""
                aws autoscaling set-desired-capacity --desired-capacity 4 --auto-scaling-group-name jenkins-
                test --region us-east-1
            """
        }
    }
}
stage("health-check"){
    steps{
        script{
            sh """
                bash -x /home/ubuntu/script.sh
            """
        }
    }
}
stage("ASG-decrease"){
    steps{
        script{
            sh"""
                aws autoscaling set-desired-capacity --desired-capacity 2 --auto-scaling-group-name jenkins-
                test --region us-east-1
            """
        }
    }
}
```

Chapter 5

Terraform

Terraform is an open source tool created by HashiCorp and written in the Go programming language.*Terraform is really easy, you just write some code and in a few minutes your infrastructure is all set up.Terraform is cloud-agnostic, write your code on one machine and run it from any machine provided that terraform is installed on it.*

Benefits of Infrastructure as Code

- Speed and safety
- Documentation
- Version control
- Validation
- Reuse

Configure Aws Credentials Example

- `Copy AWS access key id and secret access key  in your AWS credentials file`
- `You can also execute aws configure to add a new user.`

  Example

- **cat** .aws/credentials
  - **[terraform]**
  - aws_access_key_id = xxxxxxxxxxxxxxxxxx
  - aws_secret_access_key = xxx/xxxxxxxxxxxx/xxxx
- **aws configure**
  - AWS Access Key ID [****************abcd]: accesskeyid
  - AWS Secret Access Key [****************xxxx]: secret accesskey
  - Default region name [ap-south-1]: region
  - Default output format [json]:

Prerequisites

There are only two prerequisites for this:

1. Terraform – Installation instructions are covered .
2. An account on any cloud platform like AWS, Azure, GCP.

Variable.tf:-

- This file contains the variables and configurable properties used in other scripts. This file also contains the secret and access keys and key name for AWS. You will have to insert them in this file before you proceed further

Terraform

Scope

- Covered almost all services and features provided by AWS. other cloud providers and 3rd party services supported as well.

Licence and Support

- Terraform is an Open Source project. Hashicorp, the company behind Terraform, is offering support plans as well.

State Management

- Terraform is storing its state on disk. Terraform is offering remote state as well, for example, based on S3 and DynamoDB.

Verify Changes

- Terraform provides a command named plan which gives you a very detailed overview of what will be modified if you apply your blueprint.

Cloudformation

Scope

- CloudFormation covers almost all bits and parts of AWS.

Licence and Support

- CloudFormation is a service offered by AWS for free. The AWS support plans include support for CloudFormation.

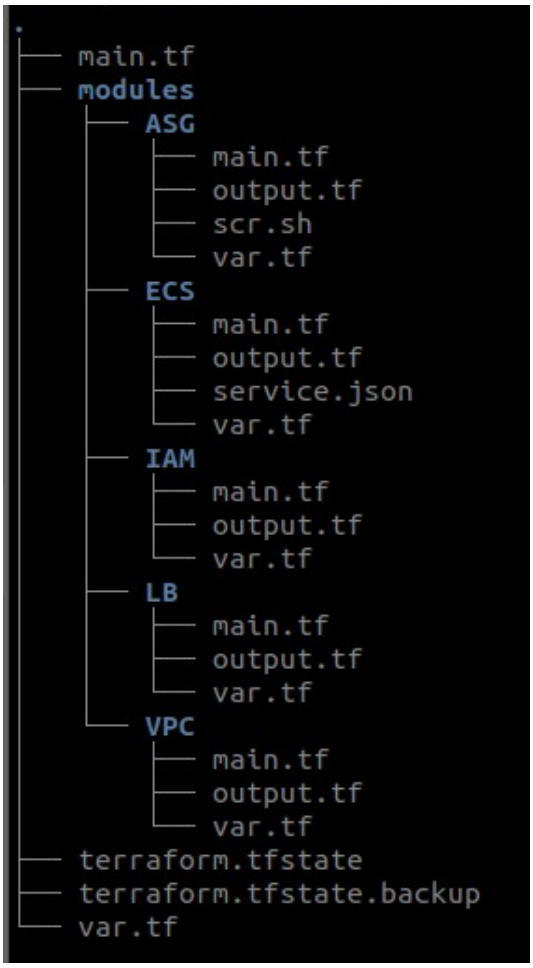
State Management

- CloudFormation is managing its state with so called stacks. By default,

Verify Changes

- CloudFormation offers change sets that you can use to verify changes

File Structure:



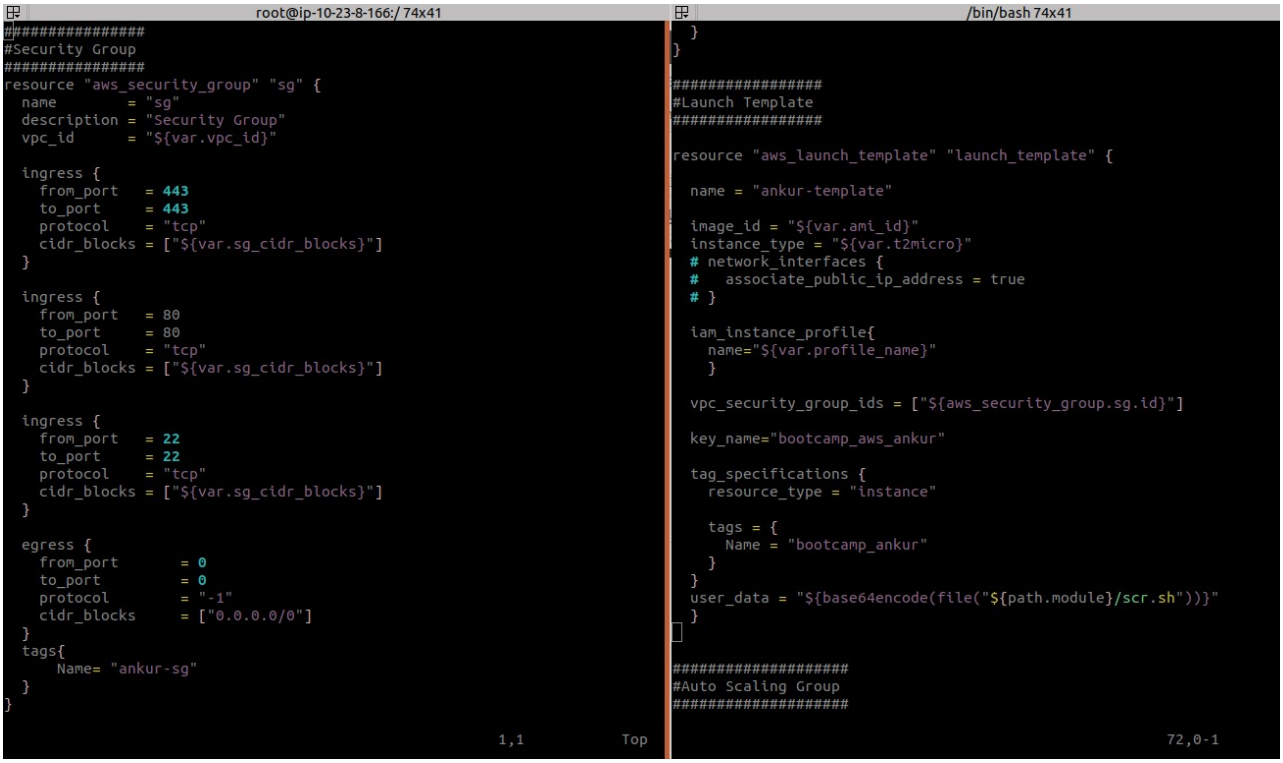
main.tf

```
provider "aws" {
    region= "${var.region}"
}

 module "my_iam"{
     source="./modules/IAM"
 }

module "my_vpc"{
    source="./modules/VPC"
}
 module "my_lb"{
     source ="./modules/LB"
     sg_lb="${module.my_asg.sg_id}"
     pub_subnet1="${module.my_vpc.Pb_Sn1_id}"
     pub_subnet2="${module.my_vpc.Pb_Sn2_id}"
     aws_vpc_id="${module.my_vpc.myvpc_id}"
 }


 module "my_asg"{
     source="./modules/ASG"
     profile_name="${module.my_iam.task_role_name}"
     vpc_id="${module.my_vpc.myvpc_id}"
     subnet_1_id="${module.my_vpc.Pb_Sn1_id}"
     subnet_2_id="${module.my_vpc.Pb_Sn2_id}"
     tg_arn1="${module.my_lb.tg_arn}"
 }
 module "my_ecs"{
     source="./modules/ECS"
     subnet_ecs1="${module.my_vpc.Pb_Sn1_id}"
     subnet_ecs2="${module.my_vpc.Pb_Sn2_id}"
     lb_arn="${module.my_lb.lb_arn}"
     tg_arn="${module.my_lb.tg_arn}"
     tg_ecs="${module.my_lb.tg_arn}"
     sg_ecs="${module.my_asg.sg_id}"
}
```

vars.tf

```
variable "region"{
     default="us-east-1"
}
~
```

./modules/ASG/main.tf




```
################
#Security Group
################
resource "aws_security_group" "sg" {
  name        = "sg"
  description = "Security Group"
  vpc_id      = "${var.vpc_id}"

  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["${var.sg_cidr_blocks}"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["${var.sg_cidr_blocks}"]
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["${var.sg_cidr_blocks}"]
  }

  egress {
    from_port       = 0
    to_port         = 0
    protocol        = "-1"
    cidr_blocks     = ["0.0.0.0/0"]
  }
  tags{
      Name= "ankur-sg"
  }
}
```

```
1,1           Top
```

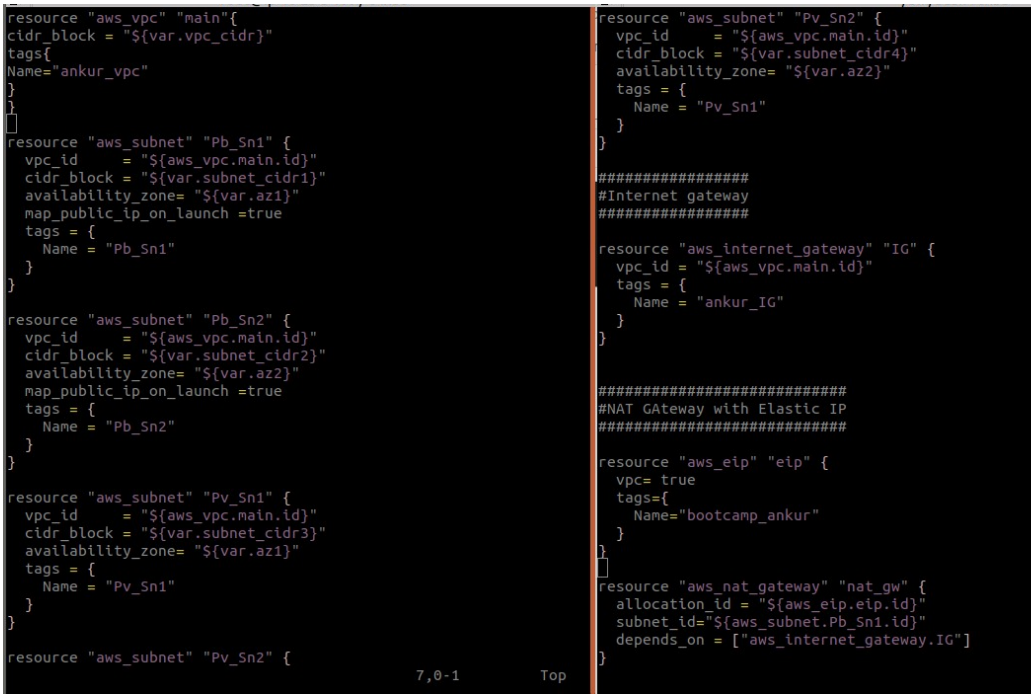
```
  }
}

#################
#Launch Template
#################

resource "aws_launch_template" "launch_template" {

  name = "ankur-template"

  image_id = "${var.ami_id}"
  instance_type = "${var.t2micro}"
  # network_interfaces {
  #   associate_public_ip_address = true
  # }

  iam_instance_profile{
    name="${var.profile_name}"
    }

  vpc_security_group_ids = ["${aws_security_group.sg.id}"]

  key_name="bootcamp_aws_ankur"

  tag_specifications {
    resource_type = "instance"

    tags = {
      Name = "bootcamp_ankur"
    }
  }
  user_data = "${base64encode(file("${path.module}/scr.sh"))}"
  }


####################
#Auto Scaling Group
####################
```

```
72,0-1
```

```
####################
#Auto Scaling Group
####################

resource "aws_autoscaling_group" "asg" {
name= "ankur-asg"
max_size=2
min_size=0
launch_template{
    id="${aws_launch_template.launch_template.id}"
    version= "$Latest"
}
health_check_type="EC2"
desired_capacity=1
vpc_zone_identifier=["${var.subnet_1_id}","${var.subnet_2_id}"]
target_group_arns=["${var.tg_arn1}"]
termination_policies=["OldestInstance"]
tag{
    key="Name"
    value= "bootcamp_ankur"
    propagate_at_launch = true
}
}
```

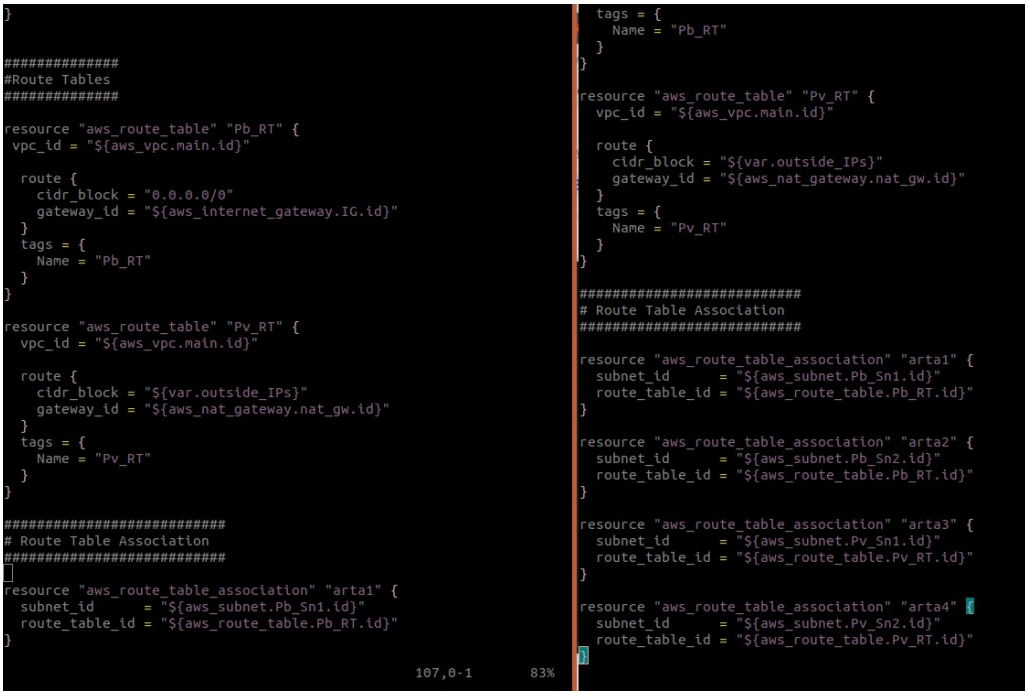
```
96,1          Bot
```

./modules/VPC/main.tf



```
resource "aws_vpc" "main"{
cidr_block = "${var.vpc_cidr}"
tags{
Name="ankur_vpc"
}
}

resource "aws_subnet" "Pb_Sn1" {
  vpc_id     = "${aws_vpc.main.id}"
  cidr_block = "${var.subnet_cidr1}"
  availability_zone= "${var.az1}"
  map_public_ip_on_launch =true
  tags = {
    Name = "Pb_Sn1"
  }
}

resource "aws_subnet" "Pb_Sn2" {
  vpc_id     = "${aws_vpc.main.id}"
  cidr_block = "${var.subnet_cidr2}"
  availability_zone= "${var.az2}"
  map_public_ip_on_launch =true
  tags = {
    Name = "Pb_Sn2"
  }
}

resource "aws_subnet" "Pv_Sn1" {
  vpc_id     = "${aws_vpc.main.id}"
  cidr_block = "${var.subnet_cidr3}"
  availability_zone= "${var.az1}"
  tags = {
    Name = "Pv_Sn1"
  }
}

resource "aws_subnet" "Pv_Sn2" {
                                    7,0-1         Top
```

```
resource "aws_subnet" "Pv_Sn2" {
  vpc_id     = "${aws_vpc.main.id}"
  cidr_block = "${var.subnet_cidr4}"
  availability_zone= "${var.az2}"
  tags = {
    Name = "Pv_Sn1"
  }
}

#################
#Internet gateway
#################

resource "aws_internet_gateway" "IG" {
  vpc_id = "${aws_vpc.main.id}"
  tags = {
    Name = "ankur_IG"
  }
}


############################
#NAT GAteway with Elastic IP
############################

resource "aws_eip" "eip" {
  vpc= true
  tags={
    Name="bootcamp_ankur"
  }
}

resource "aws_nat_gateway" "nat_gw" {
  allocation_id = "${aws_eip.eip.id}"
  subnet_id="${aws_subnet.Pb_Sn1.id}"
  depends_on = ["aws_internet_gateway.IG"]
}
```



```
}


##############
#Route Tables
##############

resource "aws_route_table" "Pb_RT" {
 vpc_id = "${aws_vpc.main.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.IG.id}"
  }
  tags = {
    Name = "Pb_RT"
  }
}

resource "aws_route_table" "Pv_RT" {
  vpc_id = "${aws_vpc.main.id}"

  route {
    cidr_block = "${var.outside_IPs}"
    gateway_id = "${aws_nat_gateway.nat_gw.id}"
  }
  tags = {
    Name = "Pv_RT"
  }
}

###########################
# Route Table Association
###########################

resource "aws_route_table_association" "arta1" {
  subnet_id      = "${aws_subnet.Pb_Sn1.id}"
  route_table_id = "${aws_route_table.Pb_RT.id}"
}
                                    107,0-1       83%
```

```
  tags = {
    Name = "Pb_RT"
  }
}

resource "aws_route_table" "Pv_RT" {
  vpc_id = "${aws_vpc.main.id}"

  route {
    cidr_block = "${var.outside_IPs}"
    gateway_id = "${aws_nat_gateway.nat_gw.id}"
  }
  tags = {
    Name = "Pv_RT"
  }
}

###########################
# Route Table Association
###########################

resource "aws_route_table_association" "arta1" {
  subnet_id      = "${aws_subnet.Pb_Sn1.id}"
  route_table_id = "${aws_route_table.Pb_RT.id}"
}

resource "aws_route_table_association" "arta2" {
  subnet_id      = "${aws_subnet.Pb_Sn2.id}"
  route_table_id = "${aws_route_table.Pb_RT.id}"
}

resource "aws_route_table_association" "arta3" {
  subnet_id      = "${aws_subnet.Pv_Sn1.id}"
  route_table_id = "${aws_route_table.Pv_RT.id}"
}

resource "aws_route_table_association" "arta4" {
  subnet_id      = "${aws_subnet.Pv_Sn2.id}"
  route_table_id = "${aws_route_table.Pv_RT.id}"
}
```

./modules/LB/main.tf

```
##############
#Target Group
##############

resource "aws_lb_target_group" "lb_tg" {
  name     = "ankur-tg"
  port     = 80
  protocol = "HTTP"
  vpc_id   = "${var.aws_vpc_id}"
}

##############
#Load Balancer
##############

resource "aws_lb" "my_lb" {
  # depends_on=["aws_lb_target_group.lb_tg"]
  name               = "ankur-lb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = ["${var.sg_lb}"]
  subnets            = ["${var.pub_subnet1}","${var.pub_subnet2}
"]

#   access_logs {
#     bucket  = "${aws_s3_bucket.lb_logs.bucket}"
#     prefix  = "ankur-lb"
#     enabled = true
#   }


    tags = {
      Name = "bootcamp_ankur"
      }
}
~
```

./modules/ECS/main.tf

```
resource "aws_lb_listener" "aws_listener" {
  load_balancer_arn = "${var.lb_arn}"
  port              = "80"
  protocol          = "HTTP"
  default_action {
    type             = "forward"
    target_group_arn = "${var.tg_arn}"
    }
}


resource "aws_ecs_cluster" "my_cluster" {
  name = "ankur-cluster"
}

resource "aws_ecs_task_definition" "ankur-td" {
  family                = "ankur-service"
  container_definitions = "${file("${path.module}/service.json")}"
  #task_role_arn =

  # volume {
  #   name      = "service-storage"
  #   host_path = "/ecs/service-storage"
  # }

  cpu=512
  memory=512

  placement_constraints {
    type       = "memberOf"
    expression = "attribute:ecs.availability-zone in [us-east-1a, us
-east-1b]"
  }
}

resource "aws_ecs_service" "my_ecs" {
  name            = "service_ankur"
  cluster         = "${aws_ecs_cluster.my_cluster.id}"
  task_definition = "${aws_ecs_task_definition.ankur-td.arn}"
  desired_count   = 1
                                                1,1           Top
```

```
  placement_constraints {
    type       = "memberOf"
    expression = "attribute:ecs.availability-zone in [us-east-1a, us-east-1b]"
  }
}

resource "aws_ecs_service" "my_ecs" {
  name            = "service_ankur"
  cluster         = "${aws_ecs_cluster.my_cluster.id}"
  task_definition = "${aws_ecs_task_definition.ankur-td.arn}"
  desired_count   = 1
  #iam_role        = "${aws_iam_role.foo.arn}"
  #depends_on      = ["aws_iam_role_policy.foo"]

  # ordered_placement_strategy {
  #   type  = "spread"
  #   field = "instanceID"
  # }

  load_balancer {
    target_group_arn = "${var.tg_ecs}"
    container_name   = "nginx"
    container_port   = 80
  }

  # network_configuration{
  #   subnets=["${var.subnet_ecs1}","${var.subnet_ecs2}"]
  #   security_groups=["${var.sg_ecs}"]
  # }
  depends_on=["aws_lb_listener.aws_listener"]
  # placement_constraints {
  #   type       = "memberOf"
  #   expression = "attribute:ecs.availability-zone in [us-west-2a, us-west-2b]"
  # }


}
~
~
                                                65,1          Bot
```

./modules/IAM/main.tf

```
resource "aws_iam_instance_profile" "Iprofile" {
  name = "Instance_Profile"
  role = "${aws_iam_role.AssumeRole.name}"
}

resource "aws_iam_role_policy_attachment" "ECS_Policy_Attach" {
  role       = "${aws_iam_role.AssumeRole.name}"
  policy_arn = "arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role"
}
resource "aws_iam_role" "AssumeRole" {

  name = "ankur-test-role"
  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Effect": "Allow",
      "Sid": ""
    }
  ]
}
EOF

  tags = {
    Name = "bootcamp_ankur"
  }
}
~
```

terraform.tfstate

```
{
    "version": 3,
    "terraform_version": "0.11.13",
    "serial": 26,
    "lineage": "635dc594-cd9a-5de4-887d-d5283170b520",
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {},
            "resources": {},
            "depends_on": []
        },
        {
            "path": [
                "root",
                "my_vpc"
            ],
            "outputs": {},
            "resources": {},
            "depends_on": []
        }
    ]
}
~
```

Chapter 6

References

1. https://docs.aws.amazon.com/

2. https://www.terraform.io/docs/index.html

3. https://jenkins.io/doc/