



[MCQs / Insem2 / CC.md](#)

 siddh34 CC Final

 History

 2 contributors



 1010 lines (660 sloc) | 23.6 KB

...

CC MCQs

- Soul purpose here is **practice**
- Very Important to visit this drive link below

[CC Drive](#)

UNIT 1 Fundamentals

Q1. The Media Access Control Sublayer resides in which OSI layer ?

- A Transport
- B Network
- C Physical
- D Data Link

Answer: Option D

Q2. Flow control is the mechanism to regulate the flow of information, so that a fast host cannot overrun a slow one. This is the function of the following OSI layer

- A All Layers
- B Physical Layer
- C Transport Layer
- D Application layer

Answer: Option A

Q3. Which layer helps to understand data representation in one form on a host to other host in their native representation?

- A. Application Layer
- B. Presentation Layer
- C. Session Layer
- D. Transport Layer

Answer: Option B

Q4. HTTP is an example of?

- A. Session Layer
- B. Presentation Layer
- C. Data Link Layer
- D. Application Layer

Answer: Option D

Q5. The network layer is responsible for carrying data from one host to another.

- A. TRUE
- B. FALSE

Answer: Option A

Q6. What is the maximum number of IP addresses that can be assigned to hosts on a local subnet that uses the 255.255.255.224 subnet mask?

- A. 14
- B. 15
- C. 16
- D. 30

Answer: Option D

Q7. You need to subnet a network that has 5 subnets, each with at least 16 hosts. Which classful subnet mask would you use?

- A. 255.255.255.192
- B. 255.255.255.224
- C. 255.255.255.240
- D. 255.255.255.248

Answer: Option B

Q8. Which of the following field in IPv4 datagram is not related to fragmentation?

- A. Flags
- B. adders
- C. offset
- D. TOS

Answer: Option D

Q9. A proxy firewall filters at the

- A. physical layer
- B. application layer
- C. data link layer
- D. network layer

Answer: Option B

Q10. A Packet filter firewall filters at the

- A. physical layer
- B. application layer
- C. data link layer
- D. network layer or transport layer

Answer: Option D

Q11. Firewalls can be of _____ kinds.

- A. 1
- B. 2

- C. 3
- D. 4

Answer: Option C

Q12. Firewall examines each _____ that are entering or leaving the internal network.

- A. emails users
- B. updates
- C. connections
- D. data packets

Answer: Option D

Q13. Packet filtering firewalls are deployed on _____

- A. routers
- B. switches
- C. hubs
- D. repeaters

Answer: A

Q14. The shell script is____

- A. File containing a series of commands
- B. File containing special symbols
- C. group of commands
- D. group of functions
- E. None of these

Answer: Option A

Q15. In the separate child shell process, the shell script executes.

- A. True
- B. False

Answer: Option A

Q16. We have to make it executable first by using ___, to run a script

- A. chmod +w
- B. chmod +r
- C. chmod +x
- D. chmod +rwx
- E. None of these

Answer: Option C

Q17. A filename is not accepted as an argument by which command?

- A. mailx
- B. ls
- C. paste
- D. cut
- E. None of these

Answer: Option A

Q18. do and the done keyword is used by which loop statements?

- A. for
- B. while
- C. for and while
- D. case

Answer: Option C

Q19. Which of the following is not a type of shell?

- A. The C Shell
- B. The Korn Shell
- C. The Bourne Shell
- D. The Perl Shell

Answer: Option D

Q20. In UNIX there are ___ major types of shells.

- A. 2

- B. 3
- C. 4
- D. 5

Answer: Option A

Q21. Which shell is the most common and best to use?

- A. Korn shell
- B. C shell
- C. Bourne shell
- D. Bash Shell

Answer: Option D

Q22. Read the Image & choose

What Bash script will correctly create these files?

touch file{1+10}.txt

touch file{1-10}.txt

touch file{1..10}.txt 

None of the above

Q23. Read the Image & choose

In Bash, what does the command below do?

cd -



It moves you to your home folder (whatever your current working directory happens to be). 



It deletes the current directory



It moves you to the directory you were previously in. 



It moves you one directory above your current working directory.

Q24. Read the Image & choose



::



done



\$\$ 

Q25. Read the Image & choose

How could you get a list of all .html files in your tree?

find *.html

find . -name *.html

find . -type html

find . -name *.html -print



Q26. Read the Image & choose

The command below will search the root filesystem for files named "finance.db". In this context, what information is being sent to /dev/null?

```
find / -name "finance.db" 1>results.txt 2>/dev/null
```

the names of files that do not match finance.db



information sent to the standard error-for example, errors that the find command displays as it runs



the names of files that match finance.db

information sent to the standard output-that is, the path to files the find command has located

Q27. Read the Image & choose

What is the output of this script?

```
#!/bin/bash
fname=john
john=thomas
echo ${!fname}
```



john 



thomas 



Syntax error



blank

Q28. Read the Image & choose

Which command is being run in this script to check if file.txt exists?

```
if [ -f file.txt ]; then
    echo "file.txt exists"
fi
```



/usr/bin/test



/usr/bin/[



the built-in [command 



/usr/bin/[[

UNIT 2 AWS

Q1. Amazon Relational Database Service is a variant of the _____ 5.1 database system.

1. Oracle
2. MySQL
3. SQL Server
4. All of the mentioned

Answer: Option 2

Q2. Which of the following statement is wrong about Amazon S3?

1. Amazon S3 is highly reliable
2. Amazon S3 provides large quantities of reliable storage that is highly protected
3. Amazon S3 is highly available
4. None of the mentioned

Answer: Option 3

Q3. What is AWS?

1. It is a collection of remote computing services also known as a cloud computing platform.
2. The popularization of the Internet actually enabled most cloud computing systems.
3. Amazon Transfer Acceleration
4. None of the mentioned

Answer: Option 1

Q4. What are key-pairs in AWS?

1. Secure login information for your virtual machines
2. Scale as per the predictable traffic patterns
3. Access management to control access to your AWS resources
4. None of the mentioned

Answer: Option 1

Q5. Why is AWS more economical than traditional data centers for applications with varying compute workloads?

1. Amazon EC2 costs are billed on a monthly basis.
2. Users retain full administrative access to their Amazon EC2 instances.
3. Amazon EC2 instances can be launched on demand when needed.
4. Users can permanently run enough instances to handle peak workloads

Answer: Option 3

Q6. Which AWS service would simplify the migration of a database to AWS?

1. AWS Storage Gateway
2. AWS Database Migration Service (AWS DMS)
3. Amazon EC2
4. Amazon AppStream 2.0

Answer: Option 2

Q7. Which AWS offering enables users to find, buy, and immediately start using software solutions in their AWS environment?

1. AWS Config
2. AWS OpsWorks
3. AWS SDK
4. AWS Marketplace

Answer: Option 4

Q8. Which AWS networking service enables a company to create a virtual network within AWS?

1. AWS Config
2. Amazon Route 53
3. AWS Direct Connect
4. Amazon Virtual Private Cloud (Amazon VPC)

Answer: Option 4

Q9. Which of the following is an AWS responsibility under the AWS shared responsibility model?

1. Configuring third-party applications
2. Maintaining physical hardware
3. Securing application access and data
4. Managing guest operating systems

Answer: Option 2

Q10. Which component of the AWS global infrastructure does Amazon CloudFront use to ensure low-latency delivery?

1. AWS Regions
2. Edge locations
3. Availability Zones
4. Virtual Private Cloud (VPC)

Answer: Option 2

Q11. How would a system administrator add an additional layer of login security to a user's AWS Management Console?

1. Use Amazon Cloud Directory
2. Audit AWS Identity and Access Management (IAM) roles
3. Enable multi-factor authentication
4. Enable AWS CloudTrail

Answer: Option 3

Q12. Which service can identify the user that made the API call when an Amazon EC2 instance is terminated?

1. AWS Trusted Advisor
2. AWS CloudTrail
3. AWS X-Ray
4. AWS Identity and Access Management (AWS IAM)

Answer: Option 2

Q13. Which service would be used to send alerts based on Amazon CloudWatch alarms?

1. Amazon Simple Notification Service (Amazon SNS)
2. AWS CloudTrail

- 3. AWS Trusted Advisor
- 4. Amazon Route 53

Answer: Option 1

Q14. Where can a user find information about prohibited actions on the AWS infrastructure?

- 1. AWS Trusted Advisor
- 2. AWS Identity and Access Management (IAM)
- 3. AWS Billing Console
- 4. AWS Acceptable Use Policy

Answer: Option 4

Q15. What kind of database is AWS DynamoDB?

- 1. NoSQL database
- 2. Relational database
- 3. Document database
- 4. None of the above

Answer: Option 1

Q16. What is AWS EC2?

- 1. AWS EC2 is a virtual server in the AWS Cloud
- 2. AWS EC2 is a resource monitoring service
- 3. AWS EC2 is a data center
- 4. AWS EC2 is a serverless compute service

Answer: Option 1

Q17. AWS Regions is part of the AWS Global Infrastructure.

- 1. False.
- 2. True.

Answer: Option 2

Q18. Which pillar of the AWS Well-Architected Framework focuses on recovering from service or infrastructure disruptions?

1. Operational excellence
2. Performance efficiency
3. Reliability

Answer: Option 3

Q19. AWS Artifact is a service that provides access to AWS security and compliance reports.

1. True
2. False

Answer: Option 1

Q20. Which AWS service helps you create users and provide them security access?

1. AWS CloudFormation
2. AWS Redshift
3. AWS Identity and Access Management (IAM)
4. AWS Direct Connect

Answer: Option 3

Q21. Can all accounts of an AWS Organization be billed with a single bill?

1. No
2. Yes

Answer: Option 2

Q22. If an instance store reboots, does the data in the instance persist?

1. Yes
2. No

Answer: Option 1

Q23. Based on application's network requests, AWS Web Application Firewall can block network traffic.

1. True
2. False

Answer: Option 1

Q24. With AWS EC2 you only pay for the compute time that you use.

1. True
2. False

Answer: Option 1

Q25. Which service helps you reduce development time and complexity of Machine Learning (ML)?

1. AWS SageMaker
2. AWS Redshift
3. AWS Elastic Beanstalk
4. AWS Lex

Answer: Option 1

Q26. For which of the following AWS resources, the Customer is responsible for the infrastructure-related security configurations?

- A. Amazon RDS
- B. Amazon DynamoDB
- C. Amazon EC2
- D. AWS Fargate

Answer: C

Q27. Which of the below-listed services is a region-based AWS service?

- A. AWS IAM
- B. Amazon EFS
- C. Amazon Route 53
- D. Amazon CloudFront

Answer: B

Q28. A company is planning to replace its physical on-premises compute servers with AWS serverless compute services. The company wants to be able to take advantage of advanced technologies quickly after the migration. Which pillar of the AWS Well-Architected Framework does this plan represent?

- A. Security
- B. Performance efficiency
- C. Operational excellence
- D. Reliability

Answer: B

Q29. Which component of the AWS global infrastructure is made up of one or more discrete data centers that have redundant power, networking, and connectivity?

- A. AWS Region
- B. Availability Zone
- C. Edge location
- D. AWS Outposts

Answer: B

Q30. A company wants to migrate a critical application to AWS. The application has a short runtime. The application is invoked by changes in data or by shifts in system state. The company needs a compute solution that maximizes operational efficiency and minimizes the cost of running the application. Which AWS solution should the company use to meet these requirements?

- A. Amazon EC2 On-Demand Instances
- B. AWS Lambda
- C. Amazon EC2 Reserved Instances
- D. Amazon EC2 Spot Instances

Answer: B

Q31. A company plans to create a data lake that uses Amazon S3. Which factor will have the MOST effect on cost?

- A. The selection of S3 storage tiers
- B. Charges to transfer existing data into Amazon S3
- C. The addition of S3 bucket policies
- D. S3 ingest fees for each request

Answer: A

Q32. Which task requires using AWS account root user credentials?

- A. Viewing billing information
- B. Changing the AWS Support plan
- C. Starting and stopping Amazon EC2 instances
- D. Opening an AWS Support case

Answer: B

Q33. What is the scope of a VPC within the AWS network?

- A. A VPC can span all Availability Zones globally.
- B. A VPC must span at least two subnets in each AWS Region.
- C. A VPC must span at least two edge locations in each AWS Region.
- D. A VPC can span all Availability Zones within an AWS Region.

Answer: D

Q34. A company needs to establish a connection between two VPCs. The VPCs are located in two different AWS Regions. The company wants to use the existing infrastructure of the VPCs for this connection. Which AWS service or feature can be used to establish this connection?

- A. AWS Client VPN
- B. VPC peering
- C. AWS Direct Connect
- D. VPC endpoints

Answer: B

Q35.Which AWS service or feature acts as a firewall for Amazon EC2 instances?

- A. Network ACL
- B. Elastic network interface
- C. Amazon VPC
- D. Security group

Answer: D

Q36. A company has a single Amazon EC2 instance. The company wants to adopt a highly available architecture. What can the company do to meet this requirement?

- A. Scale vertically to a larger EC2 instance size.
- B. Scale horizontally across multiple Availability Zones.
- C. Purchase an EC2 Dedicated Instance.
- D. Change the EC2 instance family to a compute optimized instance.

Answer: B

Q37. A company's on-premises application deployment cycle was 3-4 weeks. After migrating to the AWS Cloud, the company can deploy the application in 2-3 days. Which benefit has this company experienced by moving to the AWS Cloud?

- A. Elasticity
- B. Flexibility
- C. Agility
- D. Resilience

Answer: C

Q38. Which of the following are advantages of the AWS Cloud? (Choose two.)

- A. AWS management of user-owned infrastructure
- B. Ability to quickly change required capacity Most Voted
- C. High economies of scale Most Voted
- D. Increased deployment time to market
- E. Increased fixed expenses

Answer: B & C

Q39. A company is developing a mobile app that needs a high-performance NoSQL database. Which AWS services could the company use for this database? (Choose two.)

- A. Amazon Aurora
- B. Amazon RDS
- C. Amazon Redshift
- D. Amazon DocumentDB (with MongoDB compatibility)
- E. Amazon DynamoDB

Answer: D & E

Q40. Which AWS services are managed database services? (Choose two.)

- A. Amazon Elastic Block Store (Amazon EBS)
- B. Amazon S3
- C. Amazon RDS
- D. Amazon Elastic File System (Amazon EFS)
- E. Amazon DynamoDB

Answer: C & E

Q41. A company recently deployed an Amazon RDS instance in its VPC. The company needs to implement a stateful firewall to limit traffic to the private corporate network. Which AWS service or feature should the company use to limit network traffic directly to its RDS instance?

- A. Network ACLs
- B. Security groups
- C. AWS WAF
- D. Amazon GuardDuty

Answer: C

Q42. Which AWS service or feature identifies whether an Amazon S3 bucket or an IAM role has been shared with an external entity?

- A. AWS Service Catalog
- B. AWS Systems Manager
- C. AWS IAM Access Analyzer
- D. AWS Organizations

Answer: C

Q43. Which AWS service should be used to monitor Amazon EC2 instances for CPU and network utilization?

- A. Amazon Inspector
- B. AWS CloudTrail
- C. Amazon CloudWatch
- D. AWS Config

Answer: C

Q44. A company is launching an application in the AWS Cloud. The application will use Amazon S3 storage. A large team of researchers will have shared access to the data. The company must be able to recover data that is accidentally overwritten or deleted. Which S3 feature should the company turn on to meet this requirement?

- A. Server access logging
- B. S3 Versioning
- C. S3 Lifecycle rules
- D. Encryption in transit and at rest

Answer: B

Q45. Service control policies (SCPs) manage permissions for which of the following?

- A. Availability Zones
- B. AWS Regions
- C. AWS Organizations
- D. Edge locations

Answer: C

UNIT 3 Terraform

Q1. What is the file format of Terraform code files?

- A. JSON
- B. XML
- C. YAML
- D. HTML

Answer: Option C

Q2. Two things you need to connect to your AWS infrastructure using Terraform code (in terms of credentials)

- A. Access key ID, Secret access key and root username
- B. Access key ID and Secret access key
- C. Access key ID only
- D. Secret access key only

Answer: Option D

Q3. Resource names in Terraform are provider specific.

- A.True
- B.False

Answer: Option B

Q4. Which of the following files are processed by terraform?

- A. .tf and .tf.json
- B. .tf only
- C. .tm only
- D. all text files

Answer: Option A

Q5. Terraform is a _____ language.

- A. Objective
- B. Declarative
- C. Descriptive
- D. Functional

Answer: Option D

Q6. Terraform works based on master-less and agent-less architectural design.

- A. True
- B. False

Answer: Option A

Q7. Command to delete all the cloud resources built by previous terraform apply commands.

- A. `terraform undo`
- B. `terraform remove`
- C. `terraform destroy`
- D. `terraform delete`

Answer: Option B

Q8. Terraform is a

- A. IAC
- B. PaaS
- C. IaaS
- D. SaaS

Answer: Option A

Q9. Third-party plugins should be manually installed. Is that true?

- A. True
- B. False

Answer: Option A

Q10. Which language does terraform support from the below list?

- A. XML
- B. Javascript
- C. Hashicorp Language & JSON
- D. Plaintext

Answer: Option C

Q11. Do terraform workspaces help in adding/allowing multiple state files for a single configuration?

- A. True
- B. False

Answer: Option A

Q12. Does terraform standard backend type support remote management system?

- A. True
- B. False

Answer: Option B

Q13. Which command is used to launch terraform console?

- A. `terraform apply -config`
- B. `terraform console`
- C. `terrafrom plan`
- D. `terrafrom consul`

Answer : Option B

Q14. You have been asked to stop using static values and make code more dynamic. How can you achieve it? Select the correct option from below.

- A. Local values
- B. Input variables
- C. Depends_on
- D. Functions

Answer: Option B

Q15. Which of the following flags can be used with terraform apply command?

- A. Auto-approve
- B. Init
- C. Get
- D. Console

Answer: Option A

Q16. What are the two supported backend types in Terraform?

- A. Remote-backend
- B. Enhanced
- C. Local- backend
- D. Standard

Answer: Options B, D

Q17. Community providers are downloaded automatically using terraform init command. True or False?

- A. True
- B. False

Answer: Option A

Q18. A user wants to list all resources which are deployed using Terraform. How can this be done?

- A. `terraform state show`
- B. `terraform state list`
- C. `terraform show`
- D. `terraform show list`

Answer: Option B

Q19. Which of the following commands can be used to logout from terraform cloud?

- A. `Terraform logout`
- B. `Terraform -logout`
- C. `Terraform log out`
- D. `Terraform -log-out`

Answer: Option A

Q20. ou have the following provider configuration for AWS:

```
provider "aws" {
```

```
region = "eu-west-1"
}
provider "aws" {
alias  = "frankfurt"
region = "eu-central-1"
}
```

How do you specify an instance creation on eu-central-1 ?

- A. resource "aws_instance" "whizlabs" { provider = aws.central ... }
- B. resource "aws_instance" "whizlabs" { provider = aws.frankfurt ... }
- C. resource "aws_instance" "whizlabs" { ... }
- D. resource "aws_instance" "whizlabs" { provider = aws.west ... }

Answer: Option B

Q21. What of the following is not a source type for a module?

- A. SSH
- B. Github
- C. Bitbucket
- D. S3

Answer: Option A

Q22. What is a multicloud deployment?

- A. The possibility to run a simple .tf into multiple cloud using a single provider to deploy into multiple cloud providers
- B. The possibility to run your Terraform code using multiple cloud providers to deploy your infrastructure into multiple cloud providers
- C. The possibility to run your Terraform code using a single-global provider to deploy your infrastructure into multiple cloud providers
- D. The possibility to run your Terraform code by other tools such as Amazon Cloudformation

Answer: B

Q23. What is the name of the workspace when you execute "terraform init"?

- A. New
- B. No workspace is created

- C. Workspace
- D. Default

Answer: D

Q24. How can you delete the default workspace?

- A. `terraform workspace delete default`
- B. `terraform delete workspace default`
- C. `terraform workspace -rm default`
- D. None of the options are correct

Answer: D

Q25. How can you view the value of a particular output using the CLI? The output you want to query was declared like

```
output "ips" {  
  value = aws_instance.frontend.*.public_ip  
}
```

- A. `terraform output show`
- B. `terraform output show ips`
- C. `terraform output`
- D. `terraform output ips`

Answer: D

Q26. What benefits can provide the Infrastructure as Code for organizations?

- A. IaC can be used to deploy the latest features of Cloud Services
- B. Share and reusability of the code
- C. Blueprint of the DataCenter
- D. Versioning

Answer: B, C & D

Q27. What happens if the locking state fails when executing an operation in Terraform?

- A. Terraform will continuously apply its configuration without modifying the state, then you can execute a Terraform refresh to update the state
- B. Terraform will continuously apply its configuration and apply changes into

- the state
- C. Terraform will not continue to plan/apply any changes
 - D. Terraform will continuous and will force lock the state and will refresh the state

Answer: C

Q28. How do you force users to use a particular version of required providers in your terraform code?

- A. `terraform { required_providers { aws = { source = "hashicorp/aws" version = "3.74.1" } } }`
- B. `terraform { aws = { source = "hashicorp/aws" version ~> "3.74.1" } }`
- C. `aws = { source = "hashicorp/aws" version = "3.74.1" } }`
- D. `provider "aws" = { source = "hashicorp/aws" version = "3.74.1" }`

Answer: A

Q29. What of the following next arguments are not part of the generic meta-arguments for a provider?

- A. Alias
- B. Version
- C. Profile
- D. Region

Answers: C and D

Q30. local-exec invokes a process on the resource that is being created by Terraform.

- A. True
- B. False

Answer: B

Q31. What are the main advantages to use Terraform as the IaC tool?

- A. Manage infrastructure on multiple cloud providers
- B. Versioning
- C. Status of your infrastructure based on a State to track all the resources

- and components
- D. All of above

Answer: D

Q32. Is Terraform cloud-agostic?

- A. Yes
- B. No

Answer: A

Q33. Third-party plugins should be manually installed. Is that true?

- A. True
- B. False

Answer: A

Q34. What is the command to initialize the directory?

- A. `terraform init`
- B. `terraform apply`
- C. `terraform plan`
- D. `terraform destroy`

Answer: A

Q35. What is the command to create infrastructure?

- A. `terraform init`
- B. `terraform apply`
- C. `terraform plan`
- D. `terraform destroy`

Answer: B

Q36. What is the command to show the execution plan and not apply?

- A. `terraform init`
- B. `terraform apply`

- C. `terraform plan`
- D. `terraform destroy`

Answer: C

Q37. Terraform supports both cloud and on-premises infrastructure platforms. Is this true?

- A.True
- B.False

Answer: A

Q38. Terraform assumes an empty default configuration for any provider that is not explicitly configured. A provider block can be empty. Is this true?

- A.True
- B.False

Answer: A

Q39. Terraform CLI versions and provider versions are independent of each other. Is this true?

- A.True
- B.False

Answer: A

Q40. You are configuring aws provider and it is always recommended to hard code aws credentials in *.tf files. Is this true?

- A.True
- B.False

Answer: B

Q41. Why do we use modules for?

- A.Organize configuration

- B. Encapsulate configuration
- C. Both A & B
- D. None

Answer: C

Q42. A simple configuration consisting of a single directory with one or more .tf files is a module. Is this true?

- A. True
- B. False

Answer: A

Q43. Which of the following is the not Core Terraform workflow?

- A. Write
- B. Plan
- C. Apply
- D. Destroy

Answer: D



[Home](#) / [DevOps](#) / 50 Free Terraform Certification Exam Questions



50 Free Terraform Certification Exam Questions

DevOps / By Jeevitha TP / December 2, 2021

The Terraform Associate certification is for Cloud Engineers. You should have basic terminal skills and an understanding of on-premises and cloud architecture.

If you are new to the Terraform environment, we recommend you to practice in the demo environment and also check our [Terraform Associate practice exam](#).

This set of 50 free Terraform certification exam questions should give you a solid understanding of how Terraform associate exam is structured, the question format, and the exam pattern.

[Table of Contents](#)



0.1. What will you learn in Hashicorp Terraform Associate Exam?

1. HashiCorp Certified Terraform Certification Exam Questions

- 1.1. Domain : Use the Terraform CLI (outside of core workflow)
- 1.2. Domain : Use the Terraform CLI (outside of core workflow)
- 1.3. Domain : Understand Terraform Cloud and Enterprise capabilities
- 1.4. Domain : Understand Terraform's purpose (vs other IaC)
- 1.5. Domain : Understand Terraform basics
- 1.6. Domain : Interact with Terraform modules
- 1.7. Domain : Interact with Terraform modules
- 1.8. Domain : Navigate Terraform workflow
- 1.9. Domain : Implement and maintain state
- 1.10. Domain : Read, generate, and modify configuration
- 1.11. Domain : Read, generate, and modify configuration
- 1.12. Domain : Understand Terraform Cloud and Enterprise capabilities
- 1.13. Domain : Understand infrastructure as code (IaC) concepts
- 1.14. Domain : Understand Terraform's purpose (vs other IaC)
- 1.15. Domain : Use the Terraform CLI (outside of core workflow)
- 1.16. Domain : Interact with Terraform modules
- 1.17. Domain : Interact with Terraform modules
- 1.18. Domain : Implement and maintain state
- 1.19. Domain : Read, generate, and modify configuration
- 1.20. Domain : Understand Terraform basics
- 1.21. Domain : Understand Terraform basics
- 1.22. Domain : Understand Terraform basics
- 1.23. Domain : Understand infrastructure as code (IaC) concepts

What will you learn in Hashicorp Terraform Associate Exam?

This certification will help you learn DevOps and cloud engineering and it focuses mainly on upgrading your skills in Infrastructure as Code(IaC). Other than that you will learn the 9 following topics.

- Comprehending Infrastructure as Code (IaC)
- Assessing Terraform's purpose
- Fundamentals of Terraform
- Handling Terraform CLI (outside workflow)
- Managing Terraform Workflows

- Deploy and handle state
- Managing Terraform Modules
- Familiarity with Terraform cloud and Enterprise capabilities
- Read, produce, and alter configurations

Latest Updates 2023

Hashicorp Certified Terraform Associate 003 – coming soon

HashiCorp Certified Terraform Certification Exam Questions

Q 1. One of your colleagues is new to Terraform and wants to add a new workspace named new-hire. What command he should execute from the following?

- A. terraform workspace -new -new-hire
- B. terraform workspace new new-hire
- C. terraform workspace init new-hire
- D. terraform workspace new-hire

Option B:

terraform workspace new new-hire is the right syntax to be used whenever you want to create a new workspace.

Example :

terraform workspace new new-hire

Created and switched to workspace “new-hire”!

*Q 2. John is a newbie to Terraform and wants to enable detailed logging to find all the details
Which environment variable does he need to set?*

- A. TF_help
- B. TF_LOG
- C. TF_Debug
- D. TF_var_log

Option: B is correct

By default, Terraform does not provide detailed logging. To enable detailed logging, we have to set the environment variable TF_LOG.

By enabling TF_LOG, you can set to TRACE, INFO, WARN or ERROR, DEBUG Levels.

Q 3. Which option will you use to run provisioners that are not associated with any resources?

- A. Local-exec
- B. Null_resource
- C. Salt-masterless
- D. Remote-exec

Option B is correct

If you need to run provisioners that aren't directly associated with a specific resource, you can associate them with a null_resource.

Refer the link below for Explanation:

Q 4. Which language does terraform support from the below list?

- A. XML
- B. Javascript
- C. Hashicorp Language & JSON
- D. Plaintext

Option C is correct

Terraform supports Hashicorp Language & JSON, files ending in .tf and tf.json format.

Q 5. What is the provider version of Google Cloud being used in Terraform?

Google = “~> 1.9.0”

- A. 1.9.1
- B. 1.0.0
- C. 1.8.0

D. 1.9.2

Options A and D are correct

According to the Terraform doc, the operator `~>`(Pessimistic Constraint Operator) means only the minor (rightmost version increase) updates are accepted. Therefore, `~> 1.9.0` means the related module/provider requirement accepts 1.9.1 to 1.9.x, but not 1.10.0, and absolutely not 1.0.0 or 1.8.0.

Source: expression/version constraints in the Terraform docs.

In reference to this question, terraform is looking for any update above 1.9.0, which can be either 1.9.1 or 1.9.2. Hence, both answers are correct.

Q 6. On executing terraform plan, terraform scans the code and appends any missing argument before terraform apply.

- A. True
- B. False

Option: False

On executing terraform plan, terraform scans the code and checks for syntactical errors, missing arguments. Users need to fix these warnings before executing the code successfully.

Q 7. Do terraform workspaces help in adding/allowing multiple state files for a single configuration?

- A. True
- B. False

Option: True

Terraform workspaces allow configuring multiple state files and associating with a single configuration file

Q 8. Does terraform standard backend type support remote management system?

- A. True
- B. False

Option: False

The docs outline two **types of backends**: enhanced and standard. Enhanced **backends** are local, which is the default, and remote, which generally refers to **Terraform Cloud**. The one major feature of an enhanced **backend** is the **support** for remote operations

Q 9. Does terraform refresh command updates the state files?

- A. True
- B. False

Option: True

Yes, terraform refresh updates the state files to the latest unless there are any manual changes.

Q 10. Which command is used to launch terraform console?

- A. terraform apply -config
- B. terraform console
- C. terrafrom plan
- D. terrafrom consul

Answer : B

`terraform console [options] [dir]`

This command helps with an interactive command-line console for evaluating and experimenting with expressions

Q 11. Which of the following below helps users to deploy policy as a code?

- A. Resources
- B. Functions
- C. Sentinel
- D. Workspaces

Answer: C

Policy as code is the idea of writing code in a high-level language to manage and automate policies. By representing policies as code in text files, proven software development best practices can be adopted such as version control, automated testing, and automated deployment.

Sentinel is built around the idea and provides all the benefits of policy as code.

Q 12. You have been asked to stop using static values and make code more dynamic. How can you achieve it? Select the correct option from below.

- A. Local values
- B. Input variables
- C. Depends_on
- D. Functions

Answer: B

If we compare terraform with any conventional programming language, then,

Input Variables: Input variables are equivalent to function arguments.(Correct Answer)

Local Values: Local value are like function's temporary local variables.

Terraform Functions: The Terraform language includes a number of built-in functions that you can call from within expressions to transform and combine values. The Terraform language does not support user-defined functions, and so only the functions built in to the language are available for use.

Depends_On: It is a meta argument to signify explicit dependencies in terraform resource creation.

Q 13. Which of the following flags can be used with terraform apply command?

- A. Auto-approve
- B. Init
- C. Get
- D. Console

Answer – A

The behavior of terraform apply differs significantly depending on whether you

pass it the filename of a previously-saved plan file.

The `terraform apply` command executes the actions proposed in a Terraform plan.

-auto-approve: Skips interactive approval of the plan before applying. This option is ignored when you pass a previously-saved plan file because Terraform considers you passing the plan file as the approval and so will never prompt in that case.

-compact-warnings: This shows any warning messages in a compact form which includes only the summary messages unless the warnings are accompanied by at least one error and thus the warning text might be useful context for the errors.

-input=false: Disables all of Terraform's interactive prompts. Note that this also prevents Terraform from prompting for interactive approval of a plan, so Terraform will conservatively assume that you do not wish to apply the plan, causing the operation to fail. If you wish to run Terraform in a non-interactive context, see Running Terraform in Automation for some different approaches.

-lock=false: Disables Terraform's default behavior of attempting to take a read/write lock on the state for the duration of the operation.

-lock-timeout=DURATION: Unless locking is disabled with `-lock=false`, instructs Terraform to retry acquiring a lock for a period of time before returning an error. A duration syntax is a number followed by a time unit letter, such as "3s" for three seconds.

-no-color: Disables terminal formatting sequences in the output. Use this if you are running Terraform in a context where its output will be rendered by a system that cannot interpret terminal formatting.

-parallelism=n: Limit the number of concurrent operations as Terraform walks the graph. Defaults to 10.

For configurations using the local backend only, `terraform apply` also accepts the legacy options `-state`, `-state-out`, and `-backup`.

Q 14. What is the default number of concurrent operations supported by `terraform apply` command?

- A. 100
- B. 10
- C. 5
- D. 1

Answer – B

The default number of concurrent operations supported by Terraform apply command is 10.

-parallelism=n – Limit the number of concurrent operations as Terraform walks the graph and the default is 10

Q 15. You are trying to login into Terraform Enterprise. Which of the following command is used to save the API token?

- A. terraform get
- B. terraform API-get
- C. terraform login
- D. terraform cloud – get api

Answer – C

terraform login command can be used to automatically obtain and save an API token for Terraform Cloud, Terraform Enterprise, or any other host that offers Terraform services.

By default, Terraform will obtain an API token and save it in plain text in a local CLI configuration file called credentials.tfrc.json. When you run terraform login, it will explain specifically where it intends to save the API token and give you a chance to cancel if the current configuration is not as desired.

The syntax for Terraform login:

terraform login [hostname]

If you don't provide an explicit hostname, Terraform will assume you want to log in to Terraform Cloud at app.terraform.io.

Q 16. What are the two supported backend types in Terraform?

- A. Remote-backend
- B. Enhanced
- C. Local- backend
- D. Standard

Answer B, D

Terraform's backends are divided into two main types, according to how they handle state and operations:

- Enhanced backends can both store state and perform operations. There are only two enhanced backends: local and remote.
- Standard backends only store state and rely on the local backend for performing operations.

Q 17. Is terraform state-unlock command used to unlock the locked state file?

- A. True
- B. False

Answer: False

The correct command is given below-

```
terraform force-unlock [options] LOCK_ID [DIR]
```

The above command is used to unlock the state file.

Q 18. SMB(Server Message Block) and RDP(Remote Desktop) are supported connection types in the remote-exec provisioner. True or False?

- A. True
- B. False

Answer: False

Ssh and winrm are the supported connection types in remote-exec provisioner but not SMB and RDP.

Q 19. Community providers are downloaded automatically using terraform init command. True or False?

- A. True
- B. False

Answer: True

Any community provider can be automatically downloaded from a Terraform registry by running the terraform init command.

Community providers are installed in the same way as other providers

Q 20. By using the count meta-argument, you can scale the resources by incrementing the number.

- A. True
- B. False

Answer: True

The count is one of the reserved words. One can use count for scaling instead of repeating the resources again.

The count meta-argument accepts a whole number and creates that many instances of the resource or module.

Q 21. A user wants to list all resources which are deployed using Terraform. How can this be done?

- A. `terraform state show`
- B. `terraform state list`
- C. `terraform show`
- D. `terraform show list`

Answer: B

- **Option A is INCORRECT** because this command shows the attributes of a single resource in the Terraform state file
- **Option B is CORRECT** because the `terraform state list` command is used to list resources within a Terraform state.
- **Option C is INCORRECT** because this command will output all resources and attributes in a human-readable format.
- **Option D is INCORRECT** because this option will try to display all resources and attributes in the list file. This command takes the list as an input file for the `show` command.

The command will list all resources in the state file matching the given addresses (if any). If no addresses are given, all resources are listed.

The resources listed are sorted according to module depth order followed by alphabetical. This means that resources that are in your immediate configuration are listed first, and resources that are more deeply nested

within modules are listed last.

For complex infrastructures, the state can contain thousands of resources it can filter using the id option.

Q 22. Which among the following log command should be set to get Maximum verbosity of terraform logs?

- A. set the TF_LOG=DEBUG in environment variable
- B. set the TF_LOG=INFO in environment variable
- C. set the TF_LOG=TRACE in environment variable
- D. set the TF_LOG=WARN in environment variable

Answer: C

- **Option A is INCORRECT** because this command will not give detailed verbose information compared to trace
- **Option B is INCORRECT** because this command will just give info but will output detailed information.
- **Option C is CORRECT** because this command will output more verbose information compared all other options.
- **Option D is INCORRECT** because this option will only print warn messages.

Terraform has detailed logs which can be enabled by setting the TF_LOG environment variable to any value. This will cause detailed logs to appear on stderr.

To persist logged output you can set TF_LOG_PATH in order to force the log to always be appended to a specific file when logging is enabled. Note that even when TF_LOG_PATH is set, TF_LOG must be set in order for any logging to be enabled.

Q 23. Which among the following are not module source options?

- A. Local Path
- B. Terraform registry
- C. Bit bucket
- D. HTTP URLs
- E. BLOB storage

Answer: E

Options A, B, C, and D are **INCORRECT** because these are valid source options for a module.

Option E is CORRECT because we cannot use BLOB storage as a module source option.

The source argument in a module block tells Terraform where to find the source code for the desired child module. Terraform uses this during the module installation step of `terraform init` to download the source code to a directory on a local disk so that it can be used by other Terraform commands.

Currently following are the valid source options for a module:

- Local Paths
- Terraform Registry
- Github
- Bitbucket
- Generic Git, Mercurial repositories
- HTTP URLs
- S3 buckets
- GCS buckets

Q 24. Which of the following command can be used to syntactically check to terraform configuration before using apply or plan command?

- A. `terraform fmt`
- B. `terraform validate`
- C. `terraform show`
- D. `terraform check`

Answer: B

Option A is INCORRECT because `fmt` is used to rewrite Terraform configuration files to a canonical format and style.

Option B is CORRECT because it is used to validate the terraform configuration.

Option C is INCORRECT because the `show` is used to provide human-readable output from a state or plan file.

Option D is INCORRECT because there is no command in terraform called to `check`.

terraform validate command checks whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state. Validation requires an initialized working directory with any referenced plugins and modules installed.

Q 25. When multiple team members are working on the same state file, the state file gets locked. How to remove the lock?

- A. terraform force-unlock LOCK_ID
- B. terraform force-unlock STATE_FILE
- C. terraform unlock LOCK_ID
- D. terraform force-unlock=true

Answer: A

- Option A is CORRECT force-unlock with LOCK_ID is used to remove lock on state file.
- Option B is INCORRECT because we need to pass LOCK_ID as argument not STATE_FILE.
- Option C is INCORRECT because force-unlock is used not unlock is used to remove state lock.
- Option D is INCORRECT because force-unlock needs LOCK_ID as an argument. force-unlock manually unlock the state for the defined configuration. This will not modify your infrastructure. This command removes the lock on the state for the current configuration. The behavior of this lock is dependent on the backend being used.

Local state files cannot be unlocked by another process. Usage: terraform force-unlock LOCK_ID [DIR]

Domain : Use the Terraform CLI (outside of core workflow)

Q26 : If you want to replace a particular object even though there are no configuration changes in the code, which command from below would be best suited.

- A. `terraform destroy`
- B. `terraform taint`
- C. `terraform replace`
- D. `terraform state rm`

Correct Answer: C

Explanation

If your intent is to force replacement of a particular object even though there are no configuration changes that would require it, we recommend instead to use the `-replace` option with [terraform apply](#). For example:

```
terraform apply -replace="aws_instance.example[0]"
```

Creating a plan with the “replace” option is superior to using `terraform taint` because it will allow you to see the full effect of that change before you take any externally-visible action. When you use `terraform taint` to get a similar effect, you risk someone else on your team creating a new plan against your tainted object before you’ve had a chance to review the consequences of that change yourself.

Option A is incorrect as it would destroy the current resource and other resources that need to be created again.

Option B is incorrect because in previous versions it used to happen but as per current version of terraform, the best suited command is `terraform replace`.

Option D is incorrect because this command is used to manipulate the state file.

Reference Link: <https://www.terraform.io/docs/cli/commands/taint.html>

Domain : Use the Terraform CLI (outside of core workflow)

Q27: The data directory in terraform is used to retain data that must persist from one command to the next, so it's important to have this variable set consistently throughout all the Terraform workflow commands (starting with `terraform init`) or else Terraform may be

unable to find providers, modules, and other artifacts. Which ENVIRONMENT VARIABLE is used from below to ‘set’ per-working-directory data?

- A. TF_DATA_DIR
- B. TF_WORKSPACE
- C. TF_DATA
- D. TF_DATA_WORKSPACE

Correct Answer: A

Explanation

By default per-working-directory data is written into a .terraform subdirectory of the current directory. The TF_DATA_DIR changes the path where Terraform keeps its per-working-directory data, such as the current remote backend configuration.

Option B: For multi-environment deployment, in order to select a workspace, instead of doing *'terraform workspace select your_workspace'*, it is possible to use *TF_WORKSPACE* environment variable. It cannot be used to ‘set’ per-working-directory data.

Option C: No Such ENVIRONMENT VARIABLE exists

Option D: No Such ENVIRONMENT VARIABLE exists

Reference Link: https://www.terraform.io/docs/cli/config/environment-variables.html#tf_data_dir

Domain : Understand Terraform Cloud and Enterprise capabilities

Q28 : Which of the following commands can be used to logout from terraform cloud?

- A. Terraform logout
- B. Terraform –logout
- C. Terraform log out

- D. Terraform –log-out

Correct Answer: A

Explanation

The terraform logout command is used to remove credentials stored by terraform login. These credentials are API tokens for Terraform Cloud, Terraform Enterprise, or any other host that offers Terraform services.

Reference Link: <https://www.terraform.io/docs/cli/commands/logout.html>

Q29: Debug is the most verbose log level in Terraform.

- A. True
- B. False

Correct Answer: B

Explanation

Trace is the most verbose log level in Terraform.

You can set TF_LOG to one of the log levels TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs.

For more information, refer to the link below: <https://www.terraform.io/docs/internals/debugging.html>

Domain : Understand Terraform's purpose (vs other IaC)

Q30: What is a multicloud deployment?

- A. The possibility to run a simple .tf into multiple cloud using a single provider to deploy into multiple cloud providers
- B. The possibility to run your Terraform code using multiple cloud providers to

- deploy your infrastructure into multiple cloud providers
- C. The possibility to run your Terraform code using a single-global provider to deploy your infrastructure into multiple cloud providers
- D. The possibility to run your Terraform code by other tools such as Amazon Cloudformation

Correct Answer: B

Explanation

The idea of multi cloud deployment is to run our Terraform code, using multiple providers like AWS, GCP or Azure and deployed the infrastructure into multiple clouds in a single terraform deployment

Option A is incorrect as you have to specify different providers to deploy into multiple cloud-providers

Option C is incorrect as there isn't a single-global provider to deploy the infrastructure in multiple cloud-providers

Option D is incorrect as Terraform is not designed to be used in third party applications.

Reference: <https://www.terraform.io/intro/use-cases#multi-cloud-deployment>

Domain : Understand Terraform basics

Q31 : You have the following provider configuration for AWS:

```
provider "aws"{
  region = "eu-west-1"
}
provider "aws"{
  alias = "frankfurt"
  region = "eu-central-1"
}
```

How do you specify an instance creation on eu-central-1 ?

- A. resource "aws_instance" "whizlabs" { provider = aws.central ... }

- B. resource "aws_instance" "whizlabs" { provider = aws.frankfurt ... }
- C. resource "aws_instance" "whizlabs" { ... }
- D. resource "aws_instance" "whizlabs" { provider = aws.west ... }

Correct Answer: B

Explanation

The correct way to reference a provider is specifying the provider and the alias that have been configured on the provider configuration.

Option A is incorrect as the alias assigned into the resource doesn't exist on the provider configuration.

Option C is incorrect as if you specify a provider, this will be using the default one.

Option D is incorrect as the alias assigned into the resource doesn't exist on the provider.

Reference: <https://www.terraform.io/language/providers/configuration#selecting-alternate-provider-configurations>

Domain : Interact with Terraform modules

Q32 : What of the following is not a source type for a module?

- A. SSH
- B. Github
- C. Bitbucket
- D. S3

Correct Answer: A

Explanation

You can't use a module using a SSH source from another computer or station.

Option A is correct as SSH is not the source type for a module.

Options B, C and D are true but not the only ones. All of this sources are part of the

sources allowed:

- Local paths
- Terraform Registry
- GitHub
- Bitbucket
- Generic Git, Mercurial repositories
- HTTP URLs
- S3 buckets
- GCS buckets
- Modules in Package Subdirectories

Reference: <https://www.terraform.io/language/modules/sources>

Domain : Interact with Terraform modules

Q33 : How do you download a module configured in your Terraform code?

```
module "ecs_cluster" {  
  source = "terraform-aws-modules/ecs/aws"  
  version = "2.8.0"  
  //inputs  
}
```

- A. terraform get module ecs_cluster
- B. terraform install modules ecs_cluster
- C. terraform init
- D. terraform module init

Correct Answer: C

Explanation

During terraform init, terraform searches for module blocks and is retrieved.

Option A, B and D are incorrect as those are not valid options when initializing

plugins or modules in Terraform.

Reference: <https://www.terraform.io/cli/commands/init>

Domain : Navigate Terraform workflow

Q34 : You are a DevOps Engineer working in a CI/CD Pipeline using Jenkins. You have three stages identified: Init, Plan and Apply. After your terraform plan, you need to apply your infrastructure. In your pipeline script basically you wrote: “terraform apply”.

After triggering the pipeline you see that there was no progress and the Apply stage is waiting to confirm the changes. How can you automatically apply the changes when you type “terraform apply”?

- A. `terraform apply -auto-approve`
- B. `terraform apply -yes`
- C. `terraform apply | echo “yes”`
- D. `terraform apply | yes`

Correct Answer: A

Option A is correct. Using this flag to skip the interactive auto-approve.

Option B is incorrect as this flag doesn't exists.

Option C is incorrect as this is not part of the best practices using Terraform and also this is not interacting with terraform output.

Option D is incorrect as this will prompt a syntax error in your command line.

Reference: <https://www.terraform.io/cli/commands/apply#auto-approve>

Domain : Implement and maintain state

Q35 : What is the name of the workspace when you execute “terraform init”?

- A. New
- B. No workspace is created
- C. Workspace
- D. Default

Correct Answer: D

Explanation

When you initialize a working directory, a default workspace is created with the name “default”.

Options A, B and C are incorrect.

Reference: <https://www.terraform.io/cli/workspaces#managing-workspaces>

Domain : Implement and maintain state

Q36 : How can you delete the default workspace?

- A. terraform workspace delete default
- B. terraform delete workspace default
- C. terraform workspace -rm default
- D. None of the options are correct

Correct Answer: D

Explanation

You can't delete the default workspace.

Option A is incorrect because you can't delete the default workspace

Options B and C are incorrect because those commands have syntax failures and also you can't delete the default workspace.

Reference: <https://www.terraform.io/language/state/workspaces#using-workspaces>

Domain : Read, generate, and modify configuration

Q37 : You are a Senior DevOps Engineer and you want to provision your infrastructure Terraform code in different environments having your Terraform configuration DRY. What is the best way to do it?

*You also want to minimize the number of changes in your code
(Choose the best answer regarding best practices in Terraform and DevOps)*

- A. Have a different var files per environment and apply those files to your Terraform Configuration
- B. Have different branches in your Git repository with different var files
- C. Move out from Terraform and use Terragrunt
- D. Both A and B are correct

Correct Answer: A

Explanation

To manage and have the Terraform configuration DRY, the best way is to execute `terraform apply` and have your terraform variables in a separate file with the variables for each environment.

For example:

```
// For Development environment
```

```
terraform apply -var-file="development.tfvars"
```

```
// For Production
```

```
terraform apply -var-file="production.tfvars"
```

Option B is incorrect because having multiple branches you will need to modify the Terraform configuration in different branches. Also, this is not a good DevOps practice

Option C is incorrect as you will need to modify the templates to make your code working with Terragrunt and this will not minimize the number of changes.

Option D is incorrect as Option A is correct.

Reference: <https://www.terraform.io/language/values/variables#variable-definitions-tfvars-files>

Domain : Read, generate, and modify configuration

Q38 : How can you view the value of a particular output using the CLI? The output you want to query was declared like

```
output "ips" {
  value = aws_instance.frontend.*.public_ip
}
```

- A. terraform output show
- B. terraform output show ips
- C. terraform output
- D. terraform output ips

Correct Answer: D

Explanation

To show the value of a particular output you have to specify the command `terraform output` followed by the name of the output which you want to query:

```
$ terraform output ips
```

```
ips = [
```

```
  "54.43.114.15",
```

```
  "52.12.13.4",
```

```
  "52.4.161.69"
```

```
]
```

Options A and B are incorrect as to query an output you don't have to specify the sub-command “show”. Also this will prompt an issue.

Option C is incorrect as this will be showing all the output values and not a particular one.

Reference: <https://www.terraform.io/cli/commands/output#examples>

Domain : Understand Terraform Cloud and Enterprise capabilities

Q39: You are working with different Terraform States and you need access to the Terraform state for the organization “whizlabs” and the workspace “prod”. How will you configure the Terraform Datasource?

- A. data “terraform_remote_state” “remote_state” { backend = “remote” config = { organization = “whizlabs” workspaces = { name = “prod” } } }
- B. data “terraform_remote_state” “remote_state” { backend = “remote” organization = “whizlabs” workspaces = “prod” } }
- C. data “terraform_remote_state” “remote_state” { backend = “remote” organization = “whizlabs.prod” } }
- D. None of the above

Correct Answer: A

Explanation

A datasource “terraform_remote_state” must be configured. In this configuration, the organization and the workspaces must be set under the config block to access the remote state as a datasource.

Options B and C are incorrect as there is not “config” block for the terraform_remote_state datasource configuration

Option D is incorrect as Option A is correct.

References: <https://www.terraform.io/language/state/remote-state-data>, <https://www.terraform.io/cloud-docs/workspaces/state#data-source-configuration>

Domain : Understand infrastructure as code (IaC) concepts

Q40 : What benefits can provide the Infrastructure as Code for organizations?

- A. IaC can be used to deploy the latest features of Cloud Services
- B. Share and reusability of the code
- C. Blueprint of the DataCenter
- D. Versioning

Correct Answers: B, C and D

Explanation

IaC is the way to treat Infrastructure the same way Developers treat code.

Versioning and reusability are the most important topics on IaC. Also there is a possibility to have the datacenter snapshot via IaC.

Option A is incorrect because the latest features for the Cloud Services depend on the existing of API calls exposed and used by the Terraform providers

Reference: <https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code>

Domain : Understand Terraform's purpose (vs other IaC)

Q41 : State is a necessary requirement for Terraform to function. What of the following is not a purpose of Terraform State?

- A. Mapping to the real world
- B. Performance
- C. Syncing
- D. Security

Correct Answer: D

Explanation

State is necessary to have Terraform running. Mapping to the real world, Metadata

Performance and Syncing are part of the purpose to have a Terraform State, However, Security is not part of this purpose as not always you need to manage sensitive data and there are some options to manage those values

Options A, B and C are incorrect as those options are part of the purpose to have Terraform State

Reference: <https://www.terraform.io/language/state/purpose#purpose-of-terraform-state>

Domain : Use the Terraform CLI (outside of core workflow)

Q42 : You want to test the “split” function of Terraform locally to verify the output that the split function will be returned. What is the best approach to test this function locally?

- A. echo ‘split(“”, “foo,bar,baz”)’ | terraform console
- B. echo ‘split(“”, “foo,bar,baz”)’ | terraform plan
- C. echo ‘split(“”, “foo,bar,baz”)’ | terraform apply
- D. None of above

Correct Answer: A

Explanation

Terraform console can be used to run non-interactive scripts. For example, we can make use of the split function and pipe this function to terraform console commands.

Options B and C are incorrect as terraform plan and terraform apply are used to execute configuration written in*.tf files or json files.

Resource: <https://www.terraform.io/cli/commands/console#scripting>

Domain : Interact with Terraform modules

Q43 : In Terraform, What are modules used for?

- A. Re-use configuration
- B. Ensure best practices
- C. Encapsulate configuration
- D. All of the above

Correct Answer: D

Modules are very useful to re-use configuration, promotion best practices, encapsulate configuration and also to have a configuration organize

Options A, B and C are correct but are not matching with the best answer provided.

Reference: <https://learn.hashicorp.com/tutorials/terraform/module#what-are-modules-for>

Domain : Interact with Terraform modules

Q44 : Which of the following extensions is recognized by Terraform to fetch a module using an URL endpoint.

- A. Zip
- B. Tar.gz
- C. Tar.xz
- D. All of the above

Correct Answer: D

Explanation

All the extensions are recognized by Terraform:

zip

tar.bz2 and tbz2

tar.gz and tgz

tar.xz and txz

Reference: <https://www.terraform.io/language/modules/sources#fetching-archives-over-http>

Domain : Implement and maintain state

Q45 : What happens if the locking state fails when executing an operation in Terraform?

- A. Terraform will continuously apply its configuration without modifying the state, then you can execute a Terraform refresh to update the state
- B. Terraform will continuously apply its configuration and apply changes into the state
- C. Terraform will not continue to plan/apply any changes
- D. Terraform will continuous and will force lock the state and will refresh the state

Correct Answer: C

Explanation

If terraform fails to acquire the locking state, Terraform will not continue to apply any changes unless you specify the flag “-lock” to disable the locking state.

Options A, B and D are incorrect as terraform will not continue executing the plan or apply operations if terraform fails to acquire the state.

Reference: <https://www.terraform.io/language/state/locking#state-locking>

Domain : Read, generate, and modify configuration

Q46 : You want to assign the default value “No description set up” to a variable in your Terraform code just if a value has not been assigned on the variables.tf. If this value has content, you can assign the value to the variable. How can you perform this in your

Terraform code?

- A. `description = var.description == "null" ? "No description set up" : var.description`
- B. `description = if var.description == "null" then "No description set up" else var.description`
- C. `description = if (var.description == "null") then { "No description set up" } else { var.description }`
- D. `description = var.description == "null" : "No description set up" ? var.description`

Correct Answer: A

Conditions in terraform have the following syntax:

`condition ? true_val : false_val`

In our example:

`if var.description is null, then we assign description value "No description set up". if not, it would be var.description.`

Options B and C are incorrect as this will be a syntax error as Terraform don't understand about conditional blocks of if / else / then.

Option D is incorrect as the conditional syntax is not following the pattern:

`condition ? true_val : false_val`

Reference: <https://www.terraform.io/cli/commands/console#scripting>

Domain : Understand Terraform basics

Q47: How do you force users to use a particular version of required providers in your terraform code?

- A. `terraform { required_providers { aws = { source = "hashicorp/aws" version = "3.74.1" } } }`
- B. `terraform { aws = { source = "hashicorp/aws" version ~> "3.74.1" } }`
- C. `aws = { source = "hashicorp/aws" version = "3.74.1" } }`

D. provider "aws" = { source = "hashicorp/aws" version = "3.74.1" }

Correct Answer: A

Explanation

To configure a specific version of a provider that is required, a version must be under the block required_providers.

Option B is incorrect as it is missing the block 'required_providers'.

Option C is incorrect as there is no block called "aws".

Option D is incorrect as you have to specify the block 'required_providers' inside your terraform configuration.

Reference: <https://www.terraform.io/language/providers/configuration>

Domain : Understand Terraform basics

Q48 : What of the following next arguments are not part of the generic meta-arguments for a provider?

- A. Alias
- B. Version
- C. Profile
- D. Region

Correct Answers: C and D

Explanation

The main generic meta-arguments for a provider are alias and version. Other parameters are related to the provider configuration itself.

Options A and B are incorrect as both are part of the generic meta-arguments for a provider configuration.

References: AWS provider configuration: <https://registry.terraform.io/providers>

</hashicorp/aws/latest/docs#argument-reference>, https://www.terraform.io/language/providers/configuration?_ga=2.5999995.934997445.1642622103-536275114.1619454689#provider-configuration-1

Domain : Understand Terraform basics

Q49 : local-exec invokes a process on the resource that is being created by Terraform.

- A. True
- B. False

Correct Answer: B

False. The process is always invoked on the machine running Terraform.

Reference: <https://www.terraform.io/language/resources/provisioners/local-exec#local-exec-provisioner>

Domain : Understand infrastructure as code (IaC) concepts

Q50 : What are the main advantages to use Terraform as the IaC tool?

- A. Manage infrastructure on multiple cloud providers
- B. Versioning
- C. Status of your infrastructure based on a State to track all the resources and components
- D. All of above

Correct Answer: D

With Terraform as a tool for the IaC, you have multiple advantages, like use Terraform with multiple cloud providers, versioning your modules or even track all

the resources creation from the Terraform State

Options A, B and C are incorrect as you have to select the best possible option.

Reference: <https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code>

Conclusion:

This set of 50 free questions help you build an understanding of the exam pattern but not enough. To take the [Terraform associate exam](#) you may have to attempt some more practice questions. You can also sign up for a Terraform Demo environment in case your company is not using Terraform right now.

Reference Links:

- <https://docs.hashicorp.com/sentinel/concepts/policy-as-code/>
- <https://www.hashicorp.com/certification/terraform-associate>
- <https://www.terraform.io/docs/cli/commands/plan.html>

 [About the Author](#)

 [More from Author](#)

About Jeevitha TP

Jeevitha has a proven experience with a solid understanding of SEO activities such as content strategy, link building, and keyword strategy to increase rankings on all major search networks. Further, she works closely with the editorial and marketing teams to drive SEO in content creation and programming.

in

← Previous Post

Next Post →

Related Posts

The Best Way To Prepare For Puppet Certification

[Leave a Comment](#) / DevOps / By Dharmalingam N

Docker Certified Associate Practice Tests Launched

[Leave a Comment](#) / DevOps, News & Updates / By Pavan Gumaste

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

[Post Comment »](#)

What you're looking for?



© 2023 | Whizlabs Software Pvt. Ltd. All rights reserved.

Popular Posts

[15 Best Free Cloud Storage in 2023 – Up to 200...](#)

[New Microsoft Azure Certifications Path in 2023 \[Updated\]](#)

[Top 50 Business Analyst Interview Questions](#)

[Top 40+ Agile Scrum Interview Questions \(Updated\)](#)

[Top 25+ Fresher Java Interview Questions](#)

[Free AWS Solutions Architect Certification Exam...](#)

[Top 5 Agile Certifications in 2022 \(Updated\)](#)

[Top 50+ Azure Interview Questions and Answers \[2023\]](#)

[Top 50 Big Data Interview Questions And Answers...](#)

10 Most Popular Business Analysis Techniques



Search Medium



Published in Bachina Labs

You have **2** free member-only stories left this month. [Sign up for Medium and get an extra one](#)



Bhargav Bachina

[Follow](#)Jul 13, 2020 · 47 min read · ✨ · [Listen](#)[Save](#)

250 Practice Questions For Terraform Associate Certification

Read and Practice these questions before your exam



Terraform Certification

The Terraform Associate certification is for Cloud Engineers specializing in operations, IT, or developers who know the basic concepts and skills associated with open source HashiCorp Terraform. Candidates will be best prepared for this exam if they have professional experience using Terraform in production, but performing the exam objectives in a personal demo environment may also be sufficient.

Since this exam is multiple-choice, multiple-answer, and fill in the banks' questions, we need a lot of practice before the exam. This article helps you understand, practice, and

get you ready for the exam. *All the questions and answers are taken straight from their documentation. These are only practice questions.*

We are not going to discuss any concepts here, rather, I just want to create a bunch of practice questions for this exam based on the curriculum provided [here](#).

- *Understand infrastructure as code (IaC) concepts*
- *Understand Terraform's purpose (vs other IaC)*
- *Understand Terraform basics*
- *Use the Terraform CLI (outside of core workflow)*
- *Interact with Terraform modules*
- *Navigate Terraform workflow*
- *Implement and maintain state*
- *Read, generate, and modify the configuration*
- *Understand Terraform Cloud and Enterprise capabilities*

Understand infrastructure as code (IaC) concepts

Practice questions based on these concepts

- Explain what IaC is
- Describe the advantages of IaC patterns

1. What is Infrastructure as Code?

You write and execute the code to define, deploy, update, and destroy your infrastructure

2. What are the benefits of IaC?

a. Automation

We can bring up the servers with one script and scale up and down based on our load with the same script.

b. Reusability of the code

We can reuse the same code

c. Versioning

We can check it into version control and we get versioning. Now we can see an incremental history of who changed what, how is our infrastructure actually defined at any given point of time, and we have this transparency of documentation

IaC makes changes idempotent, consistent, repeatable, and predictable.

3. How using IaC make it easy to provision infrastructure?

IaC makes it easy to provision and apply infrastructure configurations, saving time. It standardizes workflows across different infrastructure providers (e.g., VMware, AWS, Azure, GCP, etc.) by using a common syntax across all of them.

4. What is Ideompotent in terms of IaC?

The idempotent characteristic provided by IaC tools ensures that, even if the same code is applied multiple times, the result remains the same.

5. What are Day 0 and Day 1 activities?

IaC can be applied throughout the lifecycle, both on the initial build, as well as throughout the life of the infrastructure. Commonly, these are referred to as Day 0 and Day 1 activities.

“Day 0” code provisions and configures your initial infrastructure.

“Day 1” refers to OS and application configurations you apply after you’ve initially built your infrastructure.

6. What are the use cases of Terraform?

- Heroku App Setup
- Multi-Tier Applications
- Self-Service Clusters
- Software Demos
- Disposable Environments
- Software Defined Networking
- Resource Schedulers
- Multi-Cloud Deployment

<https://www.terraform.io/intro/use-cases.html>

7. What are the advantages of Terraform?

- Platform Agnostic
- State Management
- Operator Confidence

<https://learn.hashicorp.com/terraform/getting-started/intro>

8. Where do you describe all the components or your entire datacenter so that Terraform provision those?

Configuration files ends with *.tf

9. How can Terraform build infrastructure so efficiently?

Terraform builds a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. Because of this, Terraform builds infrastructure as efficiently as possible, and operators get insight into dependencies in their infrastructure.

Understand Terraform's purpose (vs other IaC)

Practice questions based on these concepts

- Explain multi-cloud and provider-agnostic benefits
- Explain the benefits of state

10. What is multi-cloud deployment?

Provisioning your infrastructure into multiple cloud providers to increase fault-tolerance of your applications.

11. How multi-cloud deployment is useful?

By using only a single region or cloud provider, fault tolerance is limited by the availability of that provider.

Having a multi-cloud deployment allows for more graceful recovery of the loss of a region or entire provider.

12. What is cloud-agnostic in terms of provisioning tools?

cloud-agnostic and allows a single configuration to be used to manage multiple providers, and to even handle cross-cloud dependencies.

13. Is Terraform cloud-agnostic?

Yes

14. What is the use of terraform being cloud-agnostic?

It simplifies management and orchestration, helping operators build large-scale multi-cloud infrastructures.

15. What is the Terraform State?

Every time you run Terraform, it records information about what infrastructure it created in a *Terraform state file*.

By default, when you run Terraform in the folder `/some/folder`, Terraform creates the file `/some/folder/terraform.tfstate`.

This file contains a custom JSON format that records a mapping from the Terraform resources in your configuration files to the representation of those resources in the real world.

16. What is the purpose of the Terraform State?

Mapping to the Real World

Terraform requires some sort of database to map Terraform config to the real world because you can't find the same functionality in every cloud provider. You need to have some kind of mechanism to be cloud-agnostic

Metadata

Terraform must also track metadata such as resource dependencies, pointer to the provider configuration that was most recently used with the resource in situations where multiple aliased providers are present.

Performance

When running a terraform plan, Terraform must know the current state of resources in order to effectively determine the changes that it needs to make to reach your desired configuration.

For larger infrastructures, querying every resource is too slow. Many cloud providers do not provide APIs to query multiple resources at once, and the round trip time for each resource is hundreds of milliseconds. So, Terraform stores a cache of the attribute values for all resources in the state. This is the most optional feature of Terraform state and is done only as a performance improvement.

Syncing

When two people works on the same file and doing some changes to the infrastructure. Its very important for everyone to be working with the same state so that operations will be applied to the same remote objects.

<https://www.terraform.io/docs/state/purpose.html>

17. What is the name of the terraform state file?

`terraform.tfstate`

Understand Terraform basics

Practice questions based on these concepts

- Handle Terraform and provider installation and versioning
- Describe the plug-in based architecture
- Demonstrate using multiple providers
- Describe how Terraform finds and fetches providers
- Explain when to use and not use provisioners and when to use local-exec or remote-exec

18. How do you install terraform on different OS?

```
// Mac OS  
brew install terraform  
  
// Windows  
choco install terraform
```

<https://learn.hashicorp.com/terraform/getting-started/install>

19. How do you manually install terraform?

```
step 1: Download the zip file  
step 2: mv ~/Downloads/terraform /usr/local/bin/terraform
```

20. Where do you put terraform configurations so that you can configure some behaviors of Terraform itself?

The special terraform configuration block type is used to configure some behaviors of Terraform itself, such as requiring a minimum Terraform version to apply your configuration.

```
terraform {  
    # ...  
}
```

21. Only constants are allowed inside the terraform block. Is this correct?

Yes

Within a terraform block, only constant values can be used; arguments may not refer to named objects such as resources, input variables, etc, and may not use any of the Terraform language built-in functions.

22. What are the Providers?

A provider is a plugin that Terraform uses to translate the API interactions with the service. A provider is responsible for understanding API interactions and exposing resources. Because Terraform can interact with any API, you can represent almost any infrastructure type as a resource in Terraform.

<https://www.terraform.io/docs/configuration/providers.html>

23. How do you configure a Provider?

```
provider "google" {  
    project = "acme-app"  
    region  = "us-central1"  
}
```

The name given in the block header ("google" in this example) is the name of the provider to configure. Terraform associates each resource type with a provider by taking the first word of the resource type name (separated by underscores), and so the "google" provider is assumed to be the provider for the resource type name `google_compute_instance`.

The body of the block (between { and }) contains configuration arguments for the provider itself. Most arguments in this section are specified by the provider itself; in this example both `project` and

region are specific to the google provider.

24. What are the meta-arguments that are defined by Terraform itself and available for all provider blocks?

version: Constraining the allowed provider versions

alias: using the same provider with different configurations for different resources

25. What is Provider initialization and why do we need?

Each time a new provider is added to configuration -- either explicitly via a provider block or by adding a resource from that provider -- Terraform must initialize the provider before it can be used.

Initialization downloads and installs the provider's plugin so that it can later be executed.

26. How do you initialize any Provider?

Provider initialization is one of the actions of **terraform init**. Running this command will download and initialize any providers that are not already initialized.

27. When you run `terraform init` command, all the providers are installed in the current working directory. Is this true?

Providers downloaded by **terraform init** are only installed for the current working directory; other working directories can have their own installed provider versions.

Note that **terraform init** cannot automatically download providers that are not distributed by HashiCorp. See [Third-party Plugins](#) below for installation instructions.

28. How do you constrain the provider version?

To constrain the provider version as suggested, add a required_providers block inside a terraform block:

```
terraform {  
    required_providers {  
        aws = "~> 1.0"  
    }  
}
```

29. How do you upgrade to the latest acceptable version of the provider?

```
terraform init --upgrade
```

It upgrade to the latest acceptable version of each provider
This command also upgrades to the latest versions of all Terraform modules.

30. How many ways you can configure provider versions?

1. With required_providers blocks under terraform block

```
terraform {  
    required_providers {  
        aws = "~> 1.0"  
    }  
}
```

2. Provider version constraints can also be specified using a version argument within a provider block

```
provider {  
    version= "1.0"  
}
```

31. How do you configure Multiple Provider Instances?

alias

You can optionally define multiple configurations for the same

provider, and select which one to use on a per-resource or per-module basis.

32. Why do we need Multiple Provider instances?

Some of the example scenarios:

- a. multiple regions for a cloud platform
- b. targeting multiple Docker hosts
- c. multiple Consul hosts, etc.

33. How do we define multiple Provider configurations?

To include multiple configurations for a given provider, include multiple provider blocks with the same provider name, but set the alias meta-argument to an alias name to use for each additional configuration.

```
# The default provider configuration
provider "aws" {
    region = "us-east-1"
}

# Additional provider configuration for west coast region
provider "aws" {
    alias  = "west"
    region = "us-west-2"
}
```

34. How do you select alternate providers?

By default, resources use a default provider configuration inferred from the first word of the resource type name. For example, a resource of type aws_instance uses the default (un-aliased) aws provider configuration unless otherwise stated.

```
resource "aws_instance" "foo" {
    provider = aws.west

    # ...
}
```

35. What is the location of the user plugins directory?

Windows	%APPDATA%\terraform.d\plugins
All other systems	~/.terraform.d/plugins

36. Third-party plugins should be manually installed. Is that true?

True

37. The command `terraform init` cannot install third-party plugins? True or false?

True

Install third-party providers by placing their plugin executables in the user plugins directory. The user plugins directory is in one of the following locations, depending on the host operating system

Once a plugin is installed, `terraform init` can initialize it normally. You must run this command from the directory where the configuration files are located.

38. What is the naming scheme for provider plugins?

`terraform-provider-<NAME>_vX.Y.Z`

39. What is the CLI configuration File?

The CLI configuration file configures per-user settings for CLI behaviors, which apply across all Terraform working directories.

It is named either `.terraformrc` or `terraform.rc`

40. Where is the location of the CLI configuration File?

On Windows, the file must be named **terraform.rc** and placed in the relevant user's **%APPDATA%** directory.

On all other systems, the file must be named **.terraformrc** (note the leading period) and placed directly in the home directory of the relevant user.

The location of the Terraform CLI configuration file can also be specified using the **TF_CLI_CONFIG_FILE** environment variable.

41. What is Provider Plugin Cache?

By default, `terraform init` downloads plugins into a subdirectory of the working directory so that each working directory is self-contained. As a consequence, if you have multiple configurations that use the same provider then a separate copy of its plugin will be downloaded for each configuration.

Given that provider plugins can be quite large (on the order of hundreds of megabytes), this default behavior can be inconvenient for those with slow or metered Internet connections.

Therefore Terraform optionally allows the use of a local directory as a shared plugin cache, which then allows each distinct plugin binary to be downloaded only once.

42. How do you enable Provider Plugin Cache?

To enable the plugin cache, use the `plugin_cache_dir` setting in [the CLI configuration file](#).

```
plugin_cache_dir = "$HOME/.terraform.d/plugin-cache"
```

Alternatively, the **TF_PLUGIN_CACHE_DIR** environment variable can be used to enable caching or to override an existing cache directory within a particular shell session:

43. When you are using plugin cache you end up growing cache directory with different versions. Whose responsibility to clean it?

User

Terraform will never itself delete a plugin from the plugin cache once it's been placed there. Over time, as plugins are upgraded, the cache directory may grow to contain several unused versions which must be manually deleted.

44. Why do we need to initialize the directory?

When you create a new configuration – or check out an existing configuration from version control – you need to initialize the directory

```
// Example

provider "aws" {
    profile = "default"
    region  = "us-east-1"
}

resource "aws_instance" "example" {
    ami          = "ami-2757f631"
    instance_type = "t2.micro"
}
```

Initializing a configuration directory downloads and installs providers used in the configuration, which in this case is the aws provider. Subsequent commands will use local settings and data during initialization.

45. What is the command to initialize the directory?

```
terraform init
```

46. If different teams are working on the same configuration. How do you make files to have consistent formatting?

```
terraform fmt
```

This command automatically updates configurations in the current directory for easy readability and consistency.

47. If different teams are working on the same configuration. How do you make files to have syntactically valid and internally consistent?

`terraform validate`

This command will check and report errors within modules, attribute names, and value types.

Validate your configuration. If your configuration is valid, Terraform will return a success message.

48. What is the command to create infrastructure?

`terraform apply`

49. What is the command to show the execution plan and not apply?

`terraform plan`

50. How do you inspect the current state of the infrastructure applied?

`terraform show`

When you applied your configuration, Terraform wrote data into a file called `terraform.tfstate`. This file now contains the IDs and properties of the resources Terraform created so that it can manage or destroy those resources going forward.

51. If your state file is too big and you want to list the resources from your state. What is the command?

`terraform state list`

<https://learn.hashicorp.com/terraform/getting-started/build#manually-managing-state>

52. What is plug-in based architecture?

Defining additional features as plugins to your core platform or core application. This provides extensibility, flexibility and isolation

53. What are Provisioners?

If you need to do some initial setup on your instances, then provisioners let you upload files, run shell scripts, or install and trigger other software like configuration management tools, etc.

54. How do you define provisioners?

```
resource "aws_instance" "example" {
    ami           = "ami-b374d5a5"
    instance_type = "t2.micro"

    provisioner "local-exec" {
        command = "echo hello > hello.txt"
    }
}
```

Provisioner block within the resource block. Multiple provisioner blocks can be added to define multiple provisioning steps. Terraform supports multiple provisioners

<https://learn.hashicorp.com/terraform/getting-started/provision>

55. What are the types of provisioners?

local-exec
remote-exec

56. What is a local-exec provisioner and when do we use it?

The local-exec provisioner executing a command locally on your machine

running Terraform.

We use this when we need to do something on our local machine without needing any external URL

57. What is a remote-exec provisioner and when do we use it?

Another useful provisioner is remote-exec which invokes a script on a remote resource after it is created.

This can be used to run a configuration management tool, bootstrap into a cluster, etc.

58. Are provisioners runs only when the resource is created or destroyed?

Provisioners are only run when a resource is created or destroyed. Provisioners that are run while destroying are Destroy provisioners.

They are not a replacement for configuration management and changing the software of an already-running server, and are instead just meant as a way to bootstrap a server.

59. What do we need to use a remote-exec?

In order to use a remote-exec provisioner, you must choose an **ssh** or **winrm** connection in the form of a connection block within the provisioner.

Here is an example

```
provider "aws" {  
    profile = "default"  
    region  = "us-west-2"  
}  
  
resource "aws_key_pair" "example" {  
    key_name      = "examplekey"  
    public_key   = file("~/ssh/terraform.pub")  
}  
  
resource "aws_instance" "example" {  
    key_name      = aws_key_pair.example.key_name  
    ami           = "ami-04590e7389a6e577c"  
    instance_type = "t2.micro"
```

```
connection {  
    type      = "ssh"  
    user      = "ec2-user"  
    private_key = file("~/ssh/terraform")  
    host      = self.public_ip  
}  
  
provisioner "remote-exec" {  
    inline = [  
        "sudo amazon-linux-extras enable nginx1.12",  
        "sudo yum -y install nginx",  
        "sudo systemctl start nginx"  
    ]  
}  
}
```

60. When terraform mark the resources are tainted?

If a resource successfully creates but fails during provisioning, Terraform will error and mark the resource as "tainted".

A resource that is tainted has been physically created, but can't be considered safe to use since provisioning failed.

61. You applied the infrastructure with terraform apply and you have some tainted resources. You run an execution plan now what happens to those tainted resources?

When you generate your next execution plan, Terraform will not attempt to restart provisioning on the same resource because it isn't guaranteed to be safe.

Instead, Terraform will remove any tainted resources and create new resources, attempting to provision them again after creation.

<https://learn.hashicorp.com/terraform/getting-started/provision>

62. Terraform also does not automatically roll back and destroy the resource during the apply when the failure happens. Why?

Terraform also does not automatically roll back and destroy the resource during the apply when the failure happens, because that would

go against the execution plan: the execution plan would've said a resource will be created, but does not say it will ever be deleted. If you create an execution plan with a tainted resource, however, the plan will clearly state that the resource will be destroyed because it is tainted.

<https://learn.hashicorp.com/terraform/getting-started/provision>

63. How do you manually taint a resource?

terraform taint resource.id

64. Does the taint command modify the infrastructure?

terraform taint resource.id

This command will not modify infrastructure, but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next plan will show that the resource will be destroyed and recreated and the next apply will implement this change.

65. By default, provisioners that fail will also cause the Terraform apply itself to fail. Is this true?

True

66. By default, provisioners that fail will also cause the Terraform apply itself to fail. How do you change this?

The **on_failure** setting can be used to change this.

The allowed values are:

continue: Ignore the error and continue with creation or destruction.

fial: Raise an error and stop applying (the default behavior). If this is a creation provisioner, taint the resource.

```
// Example

resource "aws_instance" "web" {
  # ...

  provisioner "local-exec" {
    command  = "echo The server's IP address is ${self.private_ip}"
    on_failure = "continue"
  }
}
```

67. How do you define destroy provisioner and give an example?

You can define destroy provisioner with the parameter when

```
provisioner "remote-exec" {
  when = "destroy"

  # <...snip...>

}
```

68. How do you apply constraints for the provider versions?

The required_providers setting is a map specifying a version constraint for each provider required by your configuration.

```
terraform {
  required_providers {
    aws = ">= 2.7.0"
  }
}
```

69. What should you use to set both a lower and upper bound on versions for each provider?

```
~>

terraform {
  required_providers {
    aws = "~> 2.7.0"
  }
}
```

70. How do you try experimental features?

In releases where experimental features are available, you can enable them on a per-module basis by setting the experiments argument inside a terraform block:

```
terraform {  
  experiments = [example]  
}
```

71. When does the terraform does not recommend using provisions?

Passing data into virtual machines and other compute resources

<https://www.terraform.io/docs/provisioners/#passing-data-into-virtual-machines-and-other-compute-resources>

Running configuration management software

<https://www.terraform.io/docs/provisioners/#running-configuration-management-software>

72. Expressions in provisioner blocks cannot refer to their parent resource by name. Is this true?

True

The **self** object represents the provisioner's parent resource, and has all of that resource's attributes.

For example, use **self.public_ip** to reference an aws_instance's public_ip attribute.

73. What does this symbol version = “~> 1.0” mean when defining versions?

Any version more than 1.0 and less than 2.0

74. Terraform supports both cloud and on-premises infrastructure platforms. Is this true?

True

75. Terraform assumes an empty default configuration for any provider that is not explicitly configured. A provider block can be empty. Is this true?

True

76. How do you configure the required version of Terraform CLI can be used with your configuration?

The **required_version** setting can be used to constrain which versions of the Terraform CLI can be used with your configuration. If the running version of Terraform doesn't match the constraints specified, Terraform will produce an error and exit without taking any further actions.

77. Terraform CLI versions and provider versions are independent of each other. Is this true?

True

*78. You are configuring aws provider and it is always recommended to hard code aws credentials in *.tf files. Is this true?*

False

HashiCorp recommends that you never hard-code credentials into *.tf configuration files. We are explicitly defining the default AWS config profile here to illustrate how Terraform should access sensitive credentials.

If you leave out your AWS credentials, Terraform will automatically search for saved API credentials (for example, in ~/.aws/credentials) or IAM instance profile credentials. This is cleaner when .tf files are checked into source control or if there is more than one admin user

79. You are provisioning the infrastructure with the command `terraform apply` and you noticed one of the resources failed. How do you remove that resource without affecting the whole infrastructure?

You can taint the resource and the next apply will destroy the resource

```
terraform taint <resource.id>
```

Use the Terraform CLI (outside of core workflow)

Practice questions based on these concepts

- Given a scenario: choose when to use `terraform fmt` to format code
- Given a scenario: choose when to use `terraform taint` to taint Terraform resources
- Given a scenario: choose when to use `terraform import` to import existing infrastructure into your Terraform state
- Given a scenario: choose when to use `terraform workspace` to create workspaces
- Given a scenario: choose when to use `terraform state` to view Terraform state
- Given a scenario: choose when to enable verbose logging and what the outcome/value is

80. What is command `fmt`?

The `terraform fmt` command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability.

81. What is the recommended approach after upgrading terraform?

The canonical format may change in minor ways between Terraform versions, so after upgrading Terraform we recommend to proactively run `terraform fmt` on your modules along with any other changes you are making to adopt the new version.

82. What is the command usage?

`terraform fmt [options] [DIR]`

83. By default, `fmt` scans the current directory for configuration files. Is this true?

True

By default, `fmt` scans the current directory for configuration files. If the `dir` argument is provided then it will scan that given directory instead. If `dir` is a single dash (-) then `fmt` will read from standard input (STDIN).

84. You are formatting the configuration files and what is the flag you should use to see the differences?

`terraform fmt -diff`

85. You are formatting the configuration files and what is the flag you should use to process the subdirectories as well?

`terraform fmt -recursive`

86. You are formatting configuration files in a lot of directories and you don't want to see the list of file changes. What is the flag that you should use?

```
terraform fmt -list=false
```

87. What is the command taint?

The terraform **taint** command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply.

This command *will not* modify infrastructure, but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next plan will show that the resource will be destroyed and recreated and the next apply will implement this change.

88. What is the command usage?

```
terraform taint [options] address
```

The address argument is the address of the resource to mark as tainted. The address is in the resource address syntax syntax

89. When you are tainting a resource terraform reads the default state file `terraform.tfstate`. What is the flag you should use to read from a different path?

```
terraform taint -state=path
```

90. Give an example of tainting a single resource?

```
terraform taint aws_security_group.allow_all
```

The resource `aws_security_group.allow_all` in the module root has been marked as tainted.

91. Give an example of tainting a resource within a module?

```
terraform taint "module.couchbase.aws_instance.cb_node[9]"
```

Resource instance module.couchbase.aws_instance.cb_node[9] has been marked as tainted.

92. What is the command import?

The terraform import command is used to import existing resources into Terraform.

Terraform is able to import existing infrastructure. This allows you take resources you've created by some other means and bring it under Terraform management.

This is a great way to slowly transition infrastructure to Terraform, or to be able to be confident that you can use Terraform in the future if it potentially doesn't support every feature you need today.

93. What is the command import usage?

```
terraform import [options] ADDRESS ID
```

94. What is the default workspace name?

default

95. What are workspaces?

Each Terraform configuration has an associated backend that defines how operations are executed and where persistent data such as the Terraform state are stored.

The persistent data stored in the backend belongs to a workspace. Initially the backend has only one workspace, called "default", and thus there is only one Terraform state associated with that configuration.

Certain backends support multiple named workspaces, allowing multiple states to be associated with a single configuration.

96. What is the command to list the workspaces?

```
terraform workspace list
```

97. What is the command to create a new workspace?

```
terraform workspace new <name>
```

98. What is the command to show the current workspace?

```
terraform workspace show
```

99. What is the command to switch the workspace?

```
terraform workspace select <workspace name>
```

100. What is the command to delete the workspace?

```
terraform workspace delete <workspace name>
```

101. Can you delete the default workspace?

No. You can't ever delete default workspace

102. You are working on the different workspaces and you want to use a different number of instances based on the workspace. How do you achieve that?

```
resource "aws_instance" "example" {
```

```

count = "${terraform.workspace == "default" ? 5 : 1}"

# ... other arguments
}

```

103. You are working on the different workspaces and you want to use tags based on the workspace. How do you achieve that?

```

resource "aws_instance" "example" {
  tags = {
    Name = "web - ${terraform.workspace}"
  }

  # ... other arguments
}

```

104. You want to create a parallel, distinct copy of a set of infrastructure in order to test a set of changes before modifying the main production infrastructure. How do you achieve that?

Workspaces

105. What is the command state?

The `terraform state` command is used for advanced state management. As your Terraform usage becomes more advanced, there are some cases where you may need to modify the Terraform state. Rather than modify the state directly, the `terraform state` commands can be used in many cases instead.

<https://www.terraform.io/docs/commands/state/index.html>

106. What is the command usage?

`terraform state <subcommand> [options] [args]`

107. You are working on terraform files and you want to list all the resources. What is the command you should use?

```
terraform state list
```

108. How do you list the resources for the given name?

```
terraform state list <resource name>
```

109. What is the command that shows the attributes of a single resource in the state file?

```
terraform state show 'resource name'
```

110. How do you do debugging terraform?

Terraform has detailed logs which can be enabled by setting the **TF_LOG** environment variable to any value.

This will cause detailed logs to appear on stderr.

You can set **TF_LOG** to one of the log levels **TRACE**, **DEBUG**, **INFO**, **WARN** or **ERROR** to change the verbosity of the logs. **TRACE** is the most verbose and it is the default if **TF_LOG** is set to something other than a log level name.

To persist logged output you can set **TF_LOG_PATH** in order to force the log to always be appended to a specific file when logging is enabled.

Note that even when **TF_LOG_PATH** is set, **TF_LOG** must be set in order for any logging to be enabled.

<https://www.terraform.io/docs/internals/debugging.html>

111. If terraform crashes where should you see the logs?

```
crash.log
```

If Terraform ever crashes (a "panic" in the Go runtime), it saves a log file with the debug logs from the session as well as the panic message and backtrace to `crash.log`.

<https://www.terraform.io/docs/internals/debugging.html>

112. What is the first thing you should do when the terraform crashes?

panic message

The most interesting part of a crash log is the panic message itself and the backtrace immediately following. So the first thing to do is to search the file for panic

<https://www.terraform.io/docs/internals/debugging.html>

113. You are building infrastructure for different environments for example test and dev. How do you maintain separate states?

There are two primary methods to separate state between environments:

directories
workspaces

114. What is the difference between directory-separated and workspace-separated environments?

Directory separated environments rely on duplicate Terraform code, which may be useful if your deployments need differ, for example to test infrastructure changes in development. But they can run the risk of creating drift between the environments over time.

Workspace-separated environments use the same Terraform code but have different state files, which is useful if you want your environments to stay as similar to each other as possible, for example if you are providing development infrastructure to a team that wants to simulate running in production.

115. What is the command to pull the remote state?

terraform state pull

This command will download the state from its current location and output the raw format to stdout.

<https://www.terraform.io/docs/commands/state/pull.html>

116. What is the command used manually to upload a local state file to a remote state

terraform state push

The terraform state push command is used to manually upload a local state file to remote state. This command also works with local state.

<https://www.terraform.io/docs/commands/state/push.html>

117. The command terraform taint modifies the state file and doesn't modify the infrastructure. Is this true?

True

This command will not modify infrastructure, but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next plan will show that the resource will be destroyed and recreated and the next apply will implement this change.

118. Your team has decided to use terraform in your company and you have existing infrastructure. How do you migrate your existing resources to terraform and start using it?

You should use terraform import and modify the infrastrcuture in the terraform files and do the terraform workflow (init, plan, apply)

119. When you are working with the workspaces how do you access the current workspace in the configuration files?

`${terraform.workspace}`

120. When you are using workspaces where does the Terraform save the state file for the local state?

terraform.tfstate.d

For local state, Terraform stores the workspace states in a directory called terraform.tfstate.d.

121. When you are using workspaces where does the Terraform save the state file for the remote state?

For remote state, the workspaces are stored directly in the configured backend.

122. How do you remove items from the Terraform state?

terraform state rm 'packet_device.worker'

The **terraform state rm** command is used to remove items from the Terraform state. This command can remove single resources, single instances of a resource, entire modules, and more.

<https://www.terraform.io/docs/commands/state/rm.html>

123. How do you move the state from one source to another?

terraform state mv 'module.app' 'module.parent.module.app'

The terraform state mv command is used to move items in a Terraform state. This command can move single resources, single instances of a resource, entire modules, and more. This command can also move items to a completely different state file, enabling efficient refactoring.

<https://www.terraform.io/docs/commands/state/mv.html>

124. How do you rename a resource in the terraform state file?

```
terraform state mv 'packet_device.worker' 'packet_device.helper'
```

The above example renames the packet_device resource named worker to helper:

Interact with Terraform modules

Practice questions based on these concepts

- Contrast module source options
- Interact with module inputs and outputs
- Describe variable scope within modules/child modules
- Discover modules from the public Terraform Module Registry
- Defining module version

125. Where do you find and explore terraform Modules?

The Terraform Registry makes it simple to find and use modules.

The search query will look at module name, provider, and description to match your search terms. On the results page, filters can be used further refine search results.

126. How do you make sure that modules have stability and compatibility?

By default, only verified modules are shown in search results.

Verified modules are reviewed by HashiCorp to ensure stability and compatibility.

By using the filters, you can view unverified modules as well.

127. How do you download any modules?

You need to add any module in the configuration file like below

```
module "consul" {  
  source = "hashicorp/consul/aws"  
  version = "0.1.0"  
}
```

terraform init command will download and cache any modules referenced by a configuration.

128. What is the syntax for referencing a registry module?

<NAMESPACE>/<NAME>/<PROVIDER>

```
// for example  
module "consul" {  
  source = "hashicorp/consul/aws"  
  version = "0.1.0"  
}
```

129. What is the syntax for referencing a private registry module?

<HOSTNAME>/<NAMESPACE>/<NAME>/<PROVIDER>

```
// for example  
module "vpc" {  
  source = "app.terraform.io/example_corp/vpc/aws"  
  version = "0.9.3"  
}
```

130. The terraform recommends that all modules must follow semantic versioning. Is this true?

True

131. What is a Terraform Module?

A Terraform module is a set of Terraform configuration files in a

single directory. Even a simple configuration consisting of a single directory with one or more .tf files is a module.

132. Why do we use modules for?

- * Organize configuration
- * Encapsulate configuration
- * Re-use configuration
- * Provide consistency and ensure best practices

<https://learn.hashicorp.com/terraform/modules/modules-overview>

133. How do you call modules in your configuration?

Your configuration can use module blocks to call modules in other directories.

When Terraform encounters a module block, it loads and processes that module's configuration files.

134. How many ways you can load modules?

Local and remote modules

Modules can either be loaded from the local filesystem, or a remote source.

Terraform supports a variety of remote sources, including the Terraform Registry, most version control systems, HTTP URLs, and Terraform Cloud or Terraform Enterprise private module registries.

135. What are the best practices for using Modules?

1. Start writing your configuration with modules in mind. Even for modestly complex Terraform configurations managed by a single person, you'll find the benefits of using modules outweigh the time it takes to use them properly.
2. Use local modules to organize and encapsulate your code. Even if you

aren't using or publishing remote modules, organizing your configuration in terms of modules from the beginning will significantly reduce the burden of maintaining and updating your configuration as your infrastructure grows in complexity.

3. Use the public Terraform Registry to find useful modules. This way you can more quickly and confidently implement your configuration by relying on the work of others to implement common infrastructure scenarios.

4. Publish and share modules with your team. Most infrastructure is managed by a team of people, and modules are important way that teams can work together to create and maintain infrastructure. As mentioned earlier, you can publish modules either publicly or privately. We will see how to do this in a future guide in this series.

<https://learn.hashicorp.com/terraform/modules/modules-overview#module-best-practices>

136. What are the different source types for calling modules?

Local paths

Terraform Registry

GitHub

Generic Git, Mercurial repositories

Bitbucket

HTTP URLs

S3 buckets

GCS buckets

<https://www.terraform.io/docs/modules/sources.html>

137. What are the arguments you need for using modules in your configuration?

source and version

```
// example
module "consul" {
  source = "hashicorp/consul/aws"
  version = "0.1.0"
}
```

138. How do you set input variables for the modules?

The configuration that calls a module is responsible for setting its input values, which are passed as arguments in the module block. Aside from source and version, most of the arguments to a module block will set variable values.

On the Terraform registry page for the AWS VPC module, you will see an Inputs tab that describes all of the input variables that module supports.

For example, we have defined a lot of input variables for the modules such as ads, cidr, name, etc

main.tf

139. How do you access output variables from the modules?

You can access them by referring

`module.<MODULE NAME>.<OUTPUT NAME>`

140. Where do you put output variables in the configuration?

Module outputs are usually either passed to other parts of your configuration, or defined as outputs in your root module. You will see both uses in this guide.

Inside your configuration's directory, **outputs.tf** will need to contain:

outputs.tf

141. How do you pass input variables in the configuration?

You can define **variables.tf** in the root folder

```
variable "vpc_name" {
  description = "Name of VPC"
  type        = string
  default     = "example-vpc"
}
```

Then you can access these variables in the configuration like this

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.21.0"

  name = var.vpc_name
  cidr = var.vpc_cidr

  azs           = var.vpc_azs
  private_subnets = var.vpc_private_subnets
  public_subnets  = var.vpc_public_subnets

  enable_nat_gateway = var.vpc_enable_nat_gateway

  tags = var.vpc_tags
}
```

142. What is the child module?

A module that is called by another configuration is sometimes referred to as a "child module" of that configuration.

143. When you use local modules you don't have to do the command init or get every time there is a change in the local module. why?

When installing a local module, Terraform will instead refer directly to the source directory.

Because of this, Terraform will automatically notice changes to local modules without having to re-run `terraform init` or `terraform get`.

144. When you use remote modules what should you do if there is a change in the module?

When installing a remote module, Terraform will download it into the `.terraform` directory in your configuration's root directory.

You should initialize with `terraform init`

145. A simple configuration consisting of a single directory with one or more `.tf` files is a module. Is this true?

True

146. When using a new module for the first time, you must run either `terraform init` or `terraform get` to install the module. Is this true?

True

147. When installing the modules and where does the terraform save these modules?

```
.terraform/modules
// Example
.terraform/modules
└── ec2_instances
    └── terraform-aws-modules-terraform-aws-ec2-instance-ed6dc9
└── modules.json
└── vpc
    └── terraform-aws-modules-terraform-aws-vpc-2417f60
```

148. What is the required argument for the module?

`source`

All modules require a `source` argument, which is a meta-argument defined by Terraform CLI. Its value is either the path to a local directory of the module's configuration files, or a remote module source that Terraform should download and use. This value must be a literal string with no template sequences; arbitrary expressions are not allowed. For

more information on possible values for this argument, see [Module Sources](#).

149. What are the other optional meta-arguments along with the source when defining modules

version - (Optional) A version constraint string that specifies which versions of the referenced module are acceptable. The newest version matching the constraint will be used. version is supported only for modules retrieved from module registries.

providers - (Optional) A map whose keys are provider configuration names that are expected by child module and whose values are corresponding provider names in the calling module. This allows provider configurations to be passed explicitly to child modules. If not specified, the child module inherits all of the default (un-aliased) provider configurations from the calling module.

Navigate Terraform workflow

Practice questions based on these concepts

- Describe Terraform workflow (Write -> Plan -> Create)
- Initialize a Terraform working directory (terraform init)
- Validate a Terraform configuration (terraform validate)
- Generate and review an execution plan for Terraform (terraform plan)
- Execute changes to infrastructure with Terraform (terraform apply)
- Destroy Terraform managed infrastructure (terraform destroy)

150. What is the Core Terraform workflow?

The core Terraform workflow has three steps:

1. **Write** - Author infrastructure as code.

2. **Plan** - Preview changes before applying.
3. **Apply** - Provision reproducible infrastructure.

151. What is the workflow when you work as an Individual Practitioner?

<https://www.terraform.io/guides/core-workflow.html#working-as-an-individual-practitioner>

152. What is the workflow when you work as a team?

<https://www.terraform.io/guides/core-workflow.html#working-as-a-team>

153. What is the workflow when you work as a large organization?

<https://www.terraform.io/guides/core-workflow.html#the-core-workflow-enhanced-by-terraform-cloud>

154. What is the command init?

The terraform **init** command is used to initialize a working directory containing Terraform configuration files.

This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control.

It is safe to run this command multiple times.

155. You recently joined a team and you cloned a terraform configuration files from the version control system. What is the first command you should use?

`terraform init`

This command performs several different initialization steps in order to prepare a working directory for use.

This command is always safe to run multiple times, to bring the working directory up to date with changes in the configuration.

Though subsequent runs may give errors, this command will never delete your existing configuration or state.

If no arguments are given, the configuration in the current working directory is initialized. It is recommended to run Terraform with the current working directory set to the root directory of the configuration, and omit the DIR argument.

<https://www.terraform.io/docs/commands/init.html>

156. What is the flag you should use to upgrade modules and plugins a part of their respective installation steps?

upgrade

terraform init -upgrade

157. When you are doing initialization with terraform init, you want to skip backend initialization. What should you do?

terraform init --backend=false

158. When you are doing initialization with terraform init, you want to skip child module installation. What should you do?

terraform init --get=false

159. When you are doing initialization where do all the plugins stored?

On most operationg systems :
on Windows :

~/.terraform.d/plugins
%APPDATA%\terraform.d\plugins

160. When you are doing initialization with `terraform init`, you want to skip plugin installation. What should you do?

```
terraform init --get-plugins=false
```

Skips plugin installation. Terraform will use plugins installed in the user plugins directory, and any plugins already installed for the current working directory. If the installed plugins aren't sufficient for the configuration, init fails.

161. What does the command `terraform validate` does?

The **`terraform validate`** command validates the configuration files in a directory, referring only to the configuration and not accessing any remote services such as remote state, provider APIs, etc.

Validate runs checks that verify whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state.

It is thus primarily useful for general verification of reusable modules, including correctness of attribute names and value types.

<https://www.terraform.io/docs/commands/validate.html>

162. What does the command `plan` do?

The **`terraform plan`** command is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files.

163. What does the command `apply` do?

The **`terraform apply`** command is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a terraform plan execution plan.

<https://www.terraform.io/docs/commands/apply.html>

164. You are applying the infrastructure with the command `apply` and you don't want to do interactive approval. Which flag should you use?

`terraform apply -auto-approve`

<https://www.terraform.io/docs/commands/apply.html>

165. What does the command `destroy` do?

The **`terraform destroy`** command is used to destroy the Terraform-managed infrastructure.

166. How do you preview the behavior of the command `terraform destroy`?

`terraform plan -destroy`

167. What are implicit and explicit dependencies?

Implicit dependency:

By studying the resource attributes used in interpolation expressions, Terraform can automatically infer when one resource depends on another.

Terraform uses this dependency information to determine the correct order in which to create the different resources.

Implicit dependencies via interpolation expressions are the primary way to inform Terraform about these relationships, and should be used whenever possible.

Explicit dependency:

Sometimes there are dependencies between resources that are *not* visible to Terraform. The **`depends_on`** argument is accepted by any resource and accepts a list of resources to create *explicit dependencies* for.

168. Give an example of implicit dependency?

In the example below, the reference to `aws_instance.example.id` creates an **implicit dependency** on the `aws_instance` named `example`.

```
provider "aws" {
    profile      = "default"
    region       = "us-east-1"
}

resource "aws_instance" "example" {
    ami           = "ami-b374d5a5"
    instance_type = "t2.micro"
}

resource "aws_eip" "ip" {
    vpc = true
    instance = aws_instance.example.id
}
```

169. Give an example of explicit dependency?

In the example below, an application we will run on our EC2 instance expects to use a specific Amazon S3 bucket, but that dependency is configured inside the application code and thus not visible to Terraform. In that case, we can use `depends_on` to explicitly declare the dependency

```
resource "aws_s3_bucket" "example" {
    bucket = "some_bucket"
    acl     = "private"
}

resource "aws_instance" "example" {
    ami           = "ami-2757f631"
    instance_type = "t2.micro"

    depends_on = [aws_s3_bucket.example]
}
```

170. How do you save the execution plan?

```
terraform plan -out=tfplan
you can use that file with apply
terraform apply tfplan
```

171. You have started writing terraform configuration and you are using some sample configuration as a basis. How do you copy the example configuration into your working directory?

```
terraform init -from-module=MODULE-SOURCE
```

<https://www.terraform.io/docs/commands/init.html#copy-a-source-module>

172. What is the flag you should use with the terraform plan to get detailed on the exit codes?

```
terraform plan -detailed-exitcode
```

Return a detailed exit code when the command exits. When provided, this argument changes the exit codes and their meanings to provide more granular information about what the resulting plan contains:

- * 0 = Succeeded with empty diff (no changes)
- * 1 = Error
- * 2 = Succeeded with non-empty diff (changes present)

173. How do you target only specific resources when you run a terraform plan?

`-target=resource` - A Resource Address to target. This flag can be used multiple times. See below for more information.

174. How do you update the state prior to checking differences when you run a terraform plan?

```
terraform plan -refresh=true
```

175. The behavior of any `terraform destroy` command can be previewed at any time with an equivalent `terraform plan -destroy` command. Is this true?

True

176. You have the following file and created two resources `docker_image` and `docker_container` with the command `terraform apply` and you go to the terminal and delete the container with the command `docker rm`. You come back to your configuration and run the command again. Does terraform recreates the resource?

main.tf

Yes. Terraform creates the resource again since the execution plan says two resources and the terraform always maintains the desired state

177. You created a VM instance on AWS cloud provider with the terraform configuration and you log in AWS console and removed the instance. What does the next apply do?

It creates the instance again

178. You have the following file and created two resources docker_image and docker_container with the command `terraform plan` and you go to the terminal and delete the container with the command `docker rm`. You come back to your configuration and run the command again. What is the output of the command `plan`?

Terraform will perform the following actions:

```
# docker_container.nginx will be created
```

```
Plan: 1 to add, 0 to change, 0 to destroy.
```

Implement and maintain state

Practice questions based on these concepts

- Describe default local backend
- Outline state locking
- Handle backend authentication methods
- Describe remote state storage mechanisms and supported standard backends
- Describe the effect of Terraform refresh on state
- Describe backend block in configuration and best practices for partial configurations
- Understand secret management in state files

179. What are Backends?

A "backend" in Terraform determines how state is loaded and how an operation such as apply is executed. This abstraction enables non-local file state storage, remote execution, etc.

By default, Terraform uses the "local" backend, which is the normal behavior of Terraform

180. What is local Backend?

The local backend stores state on the local filesystem, locks that state using system APIs, and performs operations locally.

```
// Example  
  
terraform {  
    backend "local" {  
        path = "relative/path/to/terraform.tfstate"  
    }  
}
```

181. What is the default path for the local backend?

This defaults to "terraform.tfstate" relative to the root module by default.

182. What is State Locking?

If supported by your backend, Terraform will lock your state for all operations that could write state. This prevents others from acquiring the lock and potentially corrupting your state.

State locking happens automatically on all operations that could write state. You won't see any message that it is happening. If state locking fails, Terraform will not continue.

183. Does Terraform continue if state locking fails?

No.

If state locking fails, Terraform will not continue.

184. Can you disable state locking?

Yes.

You can disable state locking for most commands with the -lock flag but it is not recommended.

185. What are the types of Backend?

Standard: State management, functionality covered in State Storage & Locking

Enhanced: Everything in standard plus remote operations.

186. What are remote Backends?

Remote backends allow Terraform to use a shared storage space for state data, so any member of your team can use Terraform to manage the same infrastructure.

187. What is the benefit of using remote backend?

Remote state storage makes collaboration easier and keeps state and secret information off your local disk.

Remote state is loaded only in memory when it is used.

188. If you want to switch from using remote backend to local backend. What should you do?

If you want to move back to local state, you can remove the backend configuration block from your configuration and run **terraform init again**.

Terraform will once again ask if you want to migrate your state back to local.

189. What does the command refresh do?

The **terraform refresh** command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure.

This can be used to detect any drift from the last-known state, and to update the state file.

190. Does the command refresh modify the infrastructure?

The command **refresh** does not modify infrastructure, but does modify the state file.

If the state is changed, this may cause changes to occur during the next plan or apply.

191. How do you backup the state to the remote backend?

1. When configuring a backend for the first time (moving from no defined backend to explicitly configuring one), Terraform will give you the option to migrate your state to the new backend. This lets you adopt backends without losing any existing state.
2. To be extra careful, we always recommend manually backing up your state as well. You can do this by simply copying your `terraform.tfstate` file to another location.

192. What is a partial configuration in terms of configuring Backends?

You do not need to specify every required argument in the backend configuration. Omitting certain arguments may be desirable to avoid storing secrets, such as access keys, within the main configuration. When some or all of the arguments are omitted, we call this a ***partial configuration***.

193. What are the ways to provide remaining arguments when using partial configuration?

Interactively: Terraform will interactively ask you for the required values, unless interactive input is disabled. Terraform will not prompt for optional values.

File: A configuration file may be specified via the `init` command line. To specify a file, use the `-backend-config=PATH` option when running `terraform init`. If the file contains secrets it may be kept in a secure data store, such as `Vault`, in which case it must be downloaded to the local disk before running Terraform.

Command-line key/value pairs: Key/value pairs can be specified via the `init` command line. Note that many shells retain command-line flags in a history file, so this isn't recommended for secrets. To specify a single key/value pair, use the `-backend-config="KEY=VALUE"` option when running `terraform init`.

<https://www.terraform.io/docs/backends/config.html>

194. What is the basic requirement when using partial configuration?

When using partial configuration, Terraform requires at a minimum that an empty backend configuration is specified in one of the root Terraform configuration files, to specify the backend type

```
// Example
terraform {
  backend "consul" {}
}
```

195. Give an example of passing partial configuration with Command-line Key/Value pairs?

```
terraform init \
  -backend-config="address=demo.consul.io" \
  -backend-config="path=example_app/terraform_state" \
  -backend-config="scheme=https"
```

196. How to unconfigure a backend?

If you no longer want to use any backend, you can simply remove the configuration from the file. Terraform will detect this like any other change and prompt you to reinitialize.

As part of the reinitialization, Terraform will ask if you'd like to migrate your state back down to normal local state. Once this is complete then Terraform is back to behaving as it does by default.

197. How do you encrypt sensitive data in the state?

Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows the identity of the user requesting state and maintains a history of state changes. This can be used to control access and track activity. Terraform Enterprise also supports detailed audit logging.

The S3 backend supports encryption at rest when the encrypt option is enabled. IAM policies and logging can be used to identify any invalid access. Requests for the state go over a TLS connection.

198. Backends are completely optional. Is this true?

Backends are completely optional. You can successfully use Terraform without ever having to learn or use backends. However, they do solve pain points that afflict teams at a certain scale. If you're an individual, you can likely get away with never using backends.

199. What are the benefits of Backends?

Working in a team: Backends can store their state remotely and protect that state with locks to prevent corruption. Some backends such as Terraform Cloud even automatically store a history of all state revisions.

Keeping sensitive information off disk: State is retrieved from backends on demand and only stored in memory. If you're using a backend such as Amazon S3, the only location the state ever is persisted is in S3.

Remote operations: For larger infrastructures or certain changes, terraform apply can take a long, long time. Some backends support remote operations which enable the operation to execute remotely. You can then turn off your computer and your operation will still complete. Paired with remote state storage and locking above, this also helps in team environments.

200. Why should you be very careful with the Force unlocking the state?

Terraform has a force-unlock command to manually unlock the state if unlocking failed.

Be very careful with this command. If you unlock the state when someone else is holding the lock it could cause multiple writers. Force unlock should only be used to unlock your own lock in the situation where automatic unlocking failed.

To protect you, the force-unlock command requires a unique lock ID. Terraform will output this lock ID if unlocking fails. This lock ID acts as a nonce, ensuring that locks and unlocks target the correct lock.

201. You should only use force unlock command when automatic unlocking fails. Is this

true?

True

Read, generate, and modify the configuration

Practice questions based on these concepts

- Demonstrate the use of variables and outputs
- Describe secure secret injection best practice
- Understand the use of the collection and structural types
- Create and differentiate resource and data configuration
- Use resource addressing and resource parameters to connect resources together
- Use Terraform built-in functions to write configuration
- Configure resource using a dynamic block
- Describe built-in dependency management (order of execution based)

202. How do you define a variable?

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

This defines the region variable within your Terraform configuration.

203. How do you access the variable in the configuration?

```
// accessing a variable
```

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

204. How many ways you can assign variables in the configuration?

Command-line flags

```
terraform apply -var 'region=us-east-1'
```

From a file

To persist variable values, create a file and assign variables within this file. Create a file named `terraform.tfvars` with the following contents:

```
region = "us-east-1"  
  
terraform apply \  
-var-file="secret.tfvars" \  
-var-file="production.tfvars"
```

From environment variables

Terraform will read environment variables in the form of `TF_VAR_name` to find the value for a variable. For example, the `TF_VAR_region` variable can be set in the shell to set the region variable in Terraform.

UI input

If you execute `terraform apply` with any variable unspecified, Terraform will ask you to input the values interactively. These values are not saved, but this provides a convenient workflow when getting started with Terraform. UI input is not recommended for everyday use of Terraform.

205. Does environment variables support List and map types?

No

Environment variables can only populate string-type variables. List and map type variables must be populated via one of the other mechanisms.

206. How do you provision infrastructure in a staging environment or a production environment using the same Terraform configuration?

You can use different variable files with the same configuration

```
// Example

// For development
terraform apply -var-file="dev.tfvars"

// For test
terraform apply -var-file="test.tfvars"
```

207. How do you assign default values to variables?

If no value is assigned to a variable via any of these methods and the variable has a default key in its declaration, that value will be used for the variable.

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

208. What are the data types for the variables?

```
string
number
bool

list(<TYPE>)
set(<TYPE>)
map(<TYPE>)
object({<ATTR NAME> = <TYPE>, ... })
tuple([<TYPE>, ... ])
```

209. Give an example of data type List variables?

Lists are defined either explicitly or implicitly.

```
variable "availability_zone_names" {
  type      = list(string)
  default  = ["us-west-1a"]
}
```

210. Give an example of data type Map variables?

```

variable "region" {}
variable "amis" {
  type = map(string)
}

amis = {
  "us-east-1" = "ami-abc123"
  "us-west-2" = "ami-def456"
}

// accessing
resource "aws_instance" "example" {
  ami           = var.amis[var.region]
  instance_type = "t2.micro"
}

```

211. What is the Variable Definition Precedence?

The above mechanisms for setting variables can be used together in any combination. If the same variable is assigned multiple values, Terraform uses the *last* value it finds, overriding any previous values. Note that the same variable cannot be assigned multiple values within a single source.

Terraform loads variables in the following order, with later sources taking precedence over earlier ones:

- * Environment variables
- * The **terraform.tfvars** file, if present.
- * The **terraform.tfvars.json** file, if present.
- * Any ***.auto.tfvars** or ***.auto.tfvars.json** files, processed in lexical order of their filenames.
- * Any `-var` and `-var-file` options on the command line, in the order they are provided. (This includes variables set by a Terraform Cloud workspace.)

212. What are the output variables?

output variables as a way to organize data to be easily queried and

shown back to the Terraform user.

Outputs are a way to tell Terraform what data is important. This data is outputted when `apply` is called, and can be queried using the **`terraform output`** command.

213. Hoe do you define an output variable?

```
output "ip" {
  value = aws_eip.ip.public_ip
}
```

Multiple output blocks can be defined to specify multiple output variables.

214. How do you view outputs and queries them?

You will see the output when you run the following command
`terraform apply`

You can query the output with the following command
`terraform output ip`

215. What are the dynamic blocks?

some resource types include repeatable nested blocks in their arguments, which do not accept expressions

You can dynamically construct repeatable nested blocks like setting using a special dynamic block type, which is supported inside **resource, data, provider, and provisioner** blocks:

A dynamic block acts much like a for expression, but produces nested blocks instead of a complex typed value. It iterates over a given complex value, and generates a nested block for each element of that complex value.

<https://www.terraform.io/docs/configuration/expressions.html#dynamic-blocks>

example using dynamic blocks

216. What are the best practices for dynamic blocks?

Overuse of dynamic blocks can make configuration hard to read and maintain, so we recommend using them only when you need to hide details in order to build a clean user interface for a re-usable module.

Always write nested blocks out literally where possible.

217. What are the Built-in Functions?

The Terraform language includes a number of built-in functions that you can call from within expressions to transform and combine values.

```
max(5, 12, 9)
```

218. Does Terraform language support user-defined functions?

No

The Terraform language does not support user-defined functions, and so only the functions built in to the language are available for use.

219. What is the built-in function to change string to a number?

parseint parses the given string as a representation of an integer in the specified base and returns the resulting number. The base must be between 2 and 62 inclusive.

```
> parseint("100", 10)  
100
```

More Number Functions here

<https://www.terraform.io/docs/configuration/functions/abs.html>

220. What is the built-in function to evaluates given expression and returns a boolean whether the expression produced a result without any errors?

can

```
condition      = can(formatdate("", var.timestamp))
```

<https://www.terraform.io/docs/configuration/functions/can.html>

221. What is the built-in function to evaluates all of its argument expressions in turn and returns the result of the first one that does not produce any errors?

```
try  
locals {  
    example = try(  
        [tostring(var.example)],  
        tolist(var.example),  
    )  
}
```

222. *What is Resource Address?*

A **Resource Address** is a string that references a specific resource in a larger infrastructure. An address is made up of two parts:

[module path][resource spec]

223. *What is the Module path?*

A module path addresses a module within the tree of modules. It takes the form:

module.A.module.B.module.C...

Multiple modules in a path indicate nesting. If a module path is specified without a resource spec, the address applies to every resource within the module. If the module path is omitted, this addresses the root module.

224. *What is the Resource spec?*

A resource spec addresses a specific resource in the config. It takes the form:

resource_type.resource_name[resource index]

- * resource_type - Type of the resource being addressed.
 - * resource_name - User-defined name of the resource.
 - * [resource index] - an optional index into a resource with multiple instances, surrounded by square brace characters ([and]).
- // Examples

```
resource "aws_instance" "web" {
    # ...
    count = 4
}

aws_instance.web[3] // Refers to only last instance
aws_instance.web    // Refers to all four "web" instances.

resource "aws_instance" "web" {
    # ...
    for_each = {
        "terraform": "value1",
        "resource": "value2",
        "indexing": "value3",
        "example": "value4",
    }
}

aws_instance.web["example"] // Refers to only the "example" instance in
the config.
```

225. What are complex types and what are the collection types Terraform supports?

A *complex type* is a type that groups multiple values into a single value.

There are two categories of complex types:

collection types (for grouping similar values)

- * list(...): a sequence of values identified by consecutive whole numbers starting with zero.
- * map(...): a collection of values where each is identified by a string label.
- * set(...): a collection of unique values that do not have any secondary identifiers or ordering.

structural types (for grouping potentially dissimilar values).

- * object(...): a collection of named attributes that each have their own type.
- * tuple(...): a sequence of elements identified by consecutive whole numbers starting with zero, where each element has its own type.

226. What are the named values available and how do we refer to?

Terraform makes several kinds of named values available. Each of these names is an expression that references the associated value; you can use them as standalone expressions, or combine them with other expressions to compute new values.

- * `<RESOURCE TYPE>.<NAME>` is an object representing a managed resource of the given type and name. The attributes of the resource can be accessed using dot or square bracket notation.
- * `var.<NAME>` is the value of the input variable of the given name.
- * `local.<NAME>` is the value of the local value of the given name.
- * `module.<MODULE NAME>.<OUTPUT NAME>` is the value of the specified output value from a child module called by the current module.
- * `data.<DATA TYPE>.<NAME>` is an object representing a data resource of the given data source type and name. If the resource has the count argument set, the value is a list of objects representing its instances. If the resource has the for_each argument set, the value is a map of objects representing its instances.
- * `path.module` is the filesystem path of the module where the expression is placed.
- * `path.root` is the filesystem path of the root module of the configuration.
- * `path.cwd` is the filesystem path of the current working directory. In normal use of Terraform this is the same as `path.root`, but some advanced uses of Terraform run it from a directory other than the root module directory, causing these paths to be different.
- * `terraform.workspace` is the name of the currently selected workspace.

227. What is the built-in function that reads the contents of a file at the given path and returns them as a base64-encoded string?

`filebase64(path)`

<https://www.terraform.io/docs/configuration/functions/filebase64.html>

228. What is the built-in function that converts a timestamp into a different time format?

```
formatdate(spec, timestamp)
```

<https://www.terraform.io/docs/configuration/functions/formatdate.html>

229. What is the built-in function encodes a given value to a string using JSON syntax?

```
jsonencode({"hello": "world"})
```

<https://www.terraform.io/docs/configuration/functions/jsonencode.html>

230. What is the built-in function that calculates a full host IP address for a given host number within a given IP network address prefix?

```
> cidrhost("10.12.127.0/20", 16)
10.12.112.16
```

<https://www.terraform.io/docs/configuration/functions/cidrhost.html>

Understand Terraform Cloud and Enterprise capabilities

Practice questions based on these concepts

- Describe the benefits of Sentinel, registry, and workspaces
- Differentiate OSS and Terraform Cloud workspaces
- Summarize features of Terraform Cloud

231. What is Sentinel?

Sentinel is an embedded policy-as-code framework integrated with the HashiCorp Enterprise products. It enables fine-grained, logic-based policy decisions, and can be extended to use information from external sources.

232. What is the benefit of Sentinel?

Codifying policy removes the need for ticketing queues, without sacrificing enforcement.

One of the other benefits of Sentinel is that it also has a full testing framework.

Avoiding a ticketing workflow allows organizations to provide more self-service capabilities and end-to-end automation, minimizing the friction for developers and operators.

<https://www.hashicorp.com/blog/why-policy-as-code/>

233. What is the Private Module Registry?

Terraform Cloud's private module registry helps you share [Terraform modules](#) across your organization. It includes support for module versioning, a searchable and filterable list of available modules, and a configuration designer to help you build new workspaces faster.

234. What is the difference between public and private module registries when defined source?

The public registry uses a three-part <**NAMESPACE**>/<**MODULE NAME**>/<**PROVIDER**> format

private modules use a four-part <**HOSTNAME**>/<**ORGANIZATION**>/<**MODULE NAME**>/<**PROVIDER**> format

```
// example
module "vpc" {
  source  = "app.terraform.io/example_corp/vpc/aws"
  version = "1.0.4"
}
```

235. Where is the Terraform Module Registry available at?

<https://registry.terraform.io/>

236. *What is a workspace?*

A workspace contains everything Terraform needs to manage a given collection of infrastructure, and separate workspaces function like completely separate working directories.

237. *What are the benefits of workspaces?*

<https://www.hashicorp.com/resources/terraform-enterprise-understanding-workspaces-and-modules/>

238. *You are configuring a remote backend in the terraform cloud. You didn't create an organization before you do terraform init. Does it work?*

While the organization defined in the backend stanza **must** already exist,

239. *You are configuring a remote backend in the terraform cloud. You didn't create a workspace before you do terraform init. Does it work?*

Terraform Cloud will create it if necessary. If you opt to use a workspace that already exists, the workspace must not have any existing states.

240. *Terraform workspaces when you are working with CLI and Terraform workspaces in the Terraform cloud. Is this correct?*

If you are familiar with running Terraform using the CLI, you may have used Terraform workspaces. Terraform Cloud workspaces behave differently than Terraform CLI workspaces. Terraform CLI workspaces allow multiple state files to exist within a single directory, enabling you to use one configuration for multiple environments. Terraform Cloud workspaces contain everything needed to manage a given set of infrastructure, and function like separate working directories.

241. How do you authenticate the CLI with the terraform cloud?

Newer Versions:

1. `terraform login`
2. it will open the terraform cloud and generate the token
3. paste that token back in the CLI

https://learn.hashicorp.com/terraform/tfc/tfc_login

Older versions:

keep the following token in the CLI configuration file

```
credentials "app.terraform.io" {  
    token = "xxxxxxxx.atlasv1.zzzzzzzzzzzz"  
}
```

Terraform Certification m. DevOps cc. Cloud Computing f1. Software Development

242. You are building infrastructure on your local machine and you changed your backend to remote backend with the Terraform CLI. What should you do to migrate the state to the remote backend?

1.5K | 37

`terraform init`

Once you have authenticated the remote backend, you're ready to migrate your local state file to Terraform Cloud. To begin the migration, reinitialize. This causes Terraform to recognize your changed backend configuration.

During reinitialization, Terraform presents a prompt saying that it will copy the state file to the new backend. Enter "yes" and Terraform will migrate the state from your local machine to Terraform Cloud.

https://learn.hashicorp.com/terraform/tfc/tfc_migration#migrate-the-state-file

243. How do you configure remote backend with the terraform cloud?

Sign up for BB Tutorials & Thoughts

By Bachina Labs

You need to configure in the terraform block
Tutorials ranging from Beginner guides to advanced on Terraform, Backend, Blockchain, Docker, k8s, DevOps, Cloud, AI, ML. Thank you for subscribing and let me know if you want me cover anything? [Take a look.](#)

```
terraform {  
  backend "remote" {  
    hostname = "app.terraform.io"  
    organization = "<YOUR-ORG-NAME>"  
    name = "state-migration"  
  }  
}
```

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.

244. What is Run Triggers?

About Help Terms Privacy Terraform Cloud run triggers allow you to link workspaces so that a successful apply in a source workspace will queue a run in the workspace linked to it with a run trigger.

Get the Medium app For example, adding new subnets to your network configuration could trigger an update to your application configuration to rebalance servers across the



245. What is the benefit of Run Triggers?

When managing complex infrastructure with Terraform Cloud, organizing your configuration into different workspaces helps you to better manage and design your infrastructure.

Configuring run triggers between workspaces allows you to set up infrastructure pipelines as part of your overall deployment strategy.

246. What are the available permissions that terraform clouds can have?

Terraform Cloud teams can have read, plan, write, or admin permissions on individual workspaces.

247. Who can grant permissions on the workspaces?

Organization owners grant permissions by grouping users into teams and giving those teams privileges based on their need for access to individual workspaces.

248. Which plan do you need to manage teams on Terraform cloud?

Team Plan

249. How can you add users to an organization?

You can add users to an organization by inviting them using their email address.

Even if your team member has not signed up for Terraform Cloud yet, they can still accept the invitation and create a new account.

250. The Terraform Cloud Team plan charges you on a per-user basis. Is this true?

Yes. The Terraform Cloud Team plan is charged on a per-user basis so adding new users to your organization incurs cost.

Conclusion

The Terraform associate exam is multiple-choice, multiple answers, text-based, exam. These sample questions definitely help you prepare for the certification. I would recommend you through the documentation first and then refer to this afterward or right before the exam.

Unit-I Fundamentals

Cloud Computing and DevOps

Ms. Vidya S. Gaikwad
vidya.gaikwad@viit.ac.in
Department of Computer Engineering



BRACT's, Vishwakarma Institute of Information Technology, Pune-48

**(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)**

Contents

- **Network Fundamentals:**
- The OSI Model
- TCP vs UDP
- IP addressing & Subnetting
- Routing & Firewall
- Storage Fundamentals: Block Storage, Object Storage, File storage, SAN, NAS
- **Linux Introduction and Essential Commands:**
- Introduction
- History
- Usage
- Flavours
- **Linux Commands Shell Scripting:** Basics, Arithmetic & Logical Operations, Cron, Loops

Linux Introduction and Essential Commands

- The Linux command is a **utility of the Linux operating system**.
- All basic and advanced tasks can be done by executing commands.
- The commands are executed on the **Linux terminal**.
- The terminal is a command-line interface to interact with the system, which is similar to the command prompt in the Windows OS.
- Commands in Linux are **case-sensitive**.
- Linux provides a powerful command-line interface compared to other operating systems such as Windows and MacOS.
- We can do **basic work** and **advanced work** through its terminal.
- **Basic tasks such as creating a file, deleting a file, moving a file, and more.**
- **Advanced tasks such as administrative tasks (including package installation, user management), networking tasks (ssh connection), security tasks, and many more.**

Linux Introduction and Essential Commands

- Linux Directory Commands

1. pwd Command

- The ‘\$pwd’ command stands for ‘print working directory’ and as the name says, it displays the directory in which we are currently working.

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# █
```

Linux Introduction and Essential Commands

2) mkdir:

The ‘\$ mkdir’ stands for ‘make directory’ and it creates a new directory.

3) ls : The ‘ls’ command simply displays the contents of a directory.

```
[root@localhost ~]# mkdir vsg-cc
[root@localhost ~]# ls
VSG      dos      hello.c    hello.js   vsg          vsg-cc
[root@localhost ~]#
```

Linux Introduction and Essential Commands

4) rmdir : The '\$ rmdir' command **deletes any directory we want to delete and it stands for 'remove directory'**

```
[root@localhost ~]# rmdir vsg-cc
[root@localhost ~]# ls
VSG      dos      hello.c  hello.js  vsg
[root@localhost ~]# |
```

Linux Introduction and Essential Commands

5) c d:

- The '\$ cd' command stands for '**change directory**' and it changes your current directory to the 'newfolder' directory.

```
[root@localhost ~]# cd vsg
[root@localhost vsg]#
```

Linux Introduction and Essential Commands

6) cat command

- Concatenate, or **cat**, is one of the most frequently used Linux commands.
- It **lists, combines, and writes file content to the standard output**.
- To run the cat command, type **cat** followed by the file name and its extension.

```
[root@localhost ~]# cat newfile

Hello , welcome to VIIT
[root@localhost ~]# █
```

Linux Introduction and Essential Commands

- **cp** : This ‘\$ cp ‘ command stands for ‘copy’ and it simply copy/paste the file wherever you want to.

Linux Introduction and Essential Commands

- **c al** : The ‘\$ cal’ means **calendar** and it simply **display calendar on to your screen.**

Contents

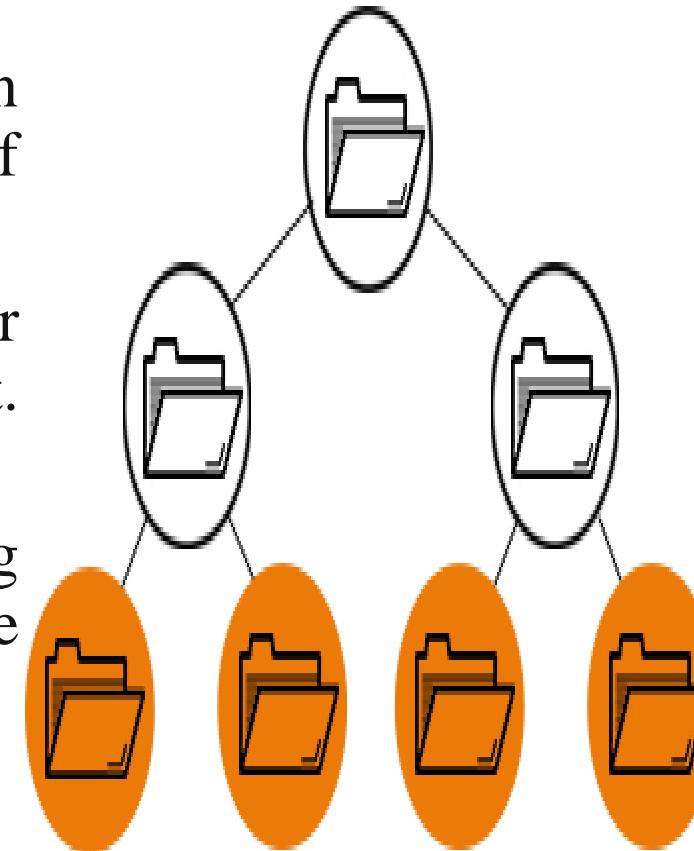
- **Network Fundamentals:**
- The OSI Model
- TCP vs UDP
- IP addressing & Subnetting
- Routing & Firewall
- **Storage Fundamentals: Block Storage, Object Storage, File storage, SAN, NAS**
- **Linux Introduction and Essential Commands:**
- Introduction
- History
- Usage
- Flavours
- **Linux Commands Shell Scripting:** Basics, Arithmetic & Logical Operations, Cron, Loops

Storage Fundamentals: Block Storage, Object Storage, File storage, SAN, NAS [1]

- Files, blocks, and objects are storage formats that hold, organize, and present data in different ways—each with their own capabilities and limitations.
- File storage organizes and represents data as a hierarchy of files in folders; block storage chunks data into arbitrarily organized, evenly sized volumes; and object storage manages data and links it to associated metadata.
- Containers are highly flexible and bring incredible scale to how apps and storage are delivered.

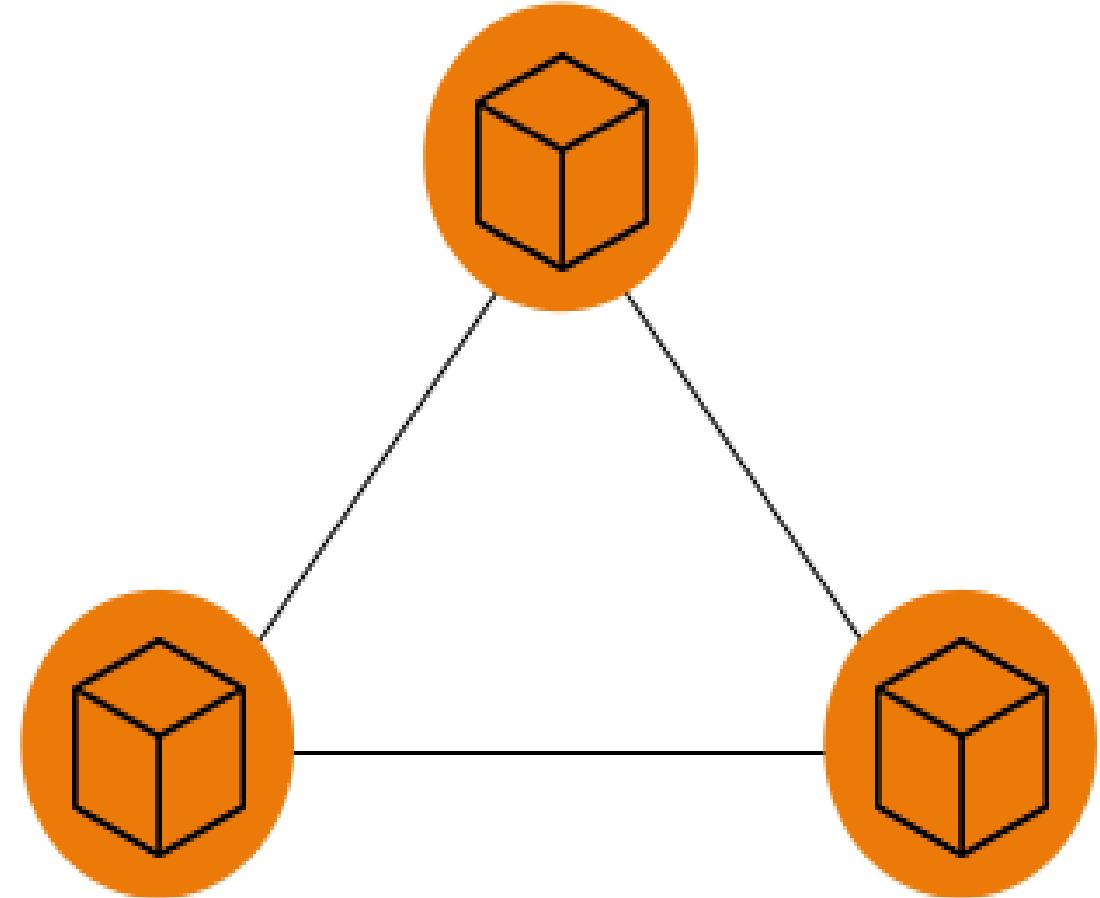
What is file storage?

- File storage, also called file-level or file-based storage.
- Data is stored as a single piece of information inside a folder, just like you'd organize pieces of paper inside a manila folder.
- When you need to access that piece of data, your computer needs to know the path to find it. (Beware—It can be a long, winding path.)
- Data stored in files is organized and retrieved using a limited amount of metadata that tells the computer exactly where the file itself is kept.
- It's like a library card catalogue for data files.



What is block storage?

- Block storage chops data into blocks—get it?—and stores them as separate pieces.
- Each block of data is given a unique identifier, which allows a storage system to place the smaller pieces of data wherever is most convenient.
- That means that some data can be stored in a [Linux®](#) environment and some can be stored in a Windows unit.

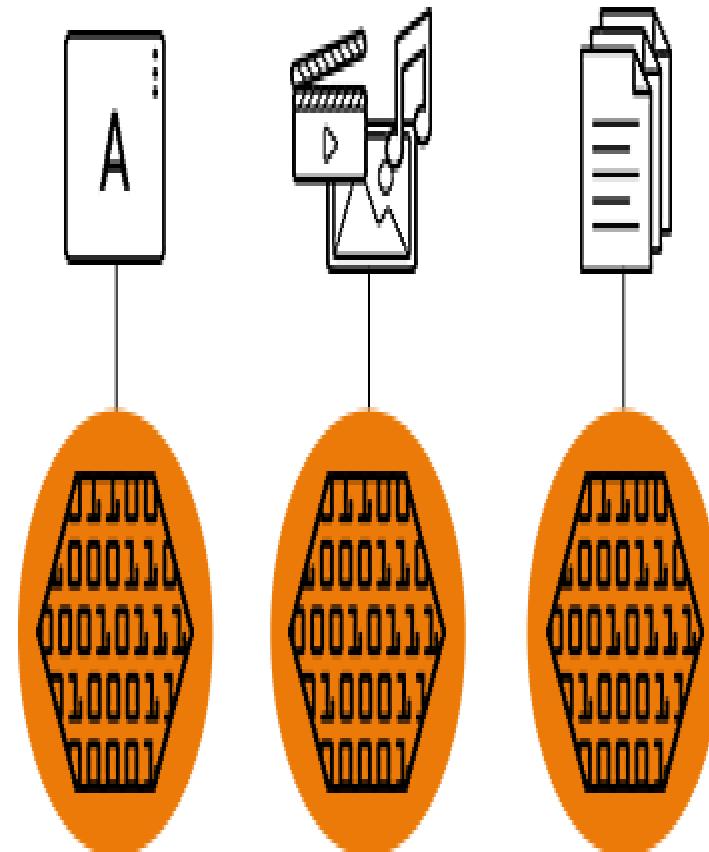


What is block storage?

- Block storage is often configured to decouple the data from the user's environment and spread it across multiple environments that can better serve the data.
- And then, when data is requested, the underlying storage software reassembles the blocks of data from these environments and presents them back to the user.
- It is usually deployed in storage-area network (SAN) environments and must be tied to a functioning server.
- Block storage doesn't rely on a single path to data—like file storage does—it can be retrieved quickly.
- Block storage can be expensive.

What is object storage?

- Object storage, also known as object-based storage, is a flat structure in which files are broken into pieces and spread out among hardware.
- In object storage, the data is broken into discrete units called objects and is kept in a single repository, instead of being kept as files in folders or as blocks on servers.
- Object storage volumes work as modular units: each is a self-contained repository that owns the data, a unique identifier that allows the object to be found over a distributed system, and the metadata that describes the data.



What is object storage?

- To retrieve the data, the storage operating system uses the metadata and identifiers, which distributes the load better and lets administrators apply policies that perform more robust searches.
- Object storage requires a simple HTTP application programming interface (API), which is used by most clients in all languages.
- Object storage is cost efficient: you only pay for what you use.
- It can scale easily, making it a great choice for public cloud storage.
- It's a storage system well suited for static data, and its agility and flat nature means it can scale to extremely large quantities of data.
- The objects have enough information for an application to find the data quickly and are good at storing unstructured data.

Capability	Object Storage	File Storage	Block Storage
Consistency	Eventual consistency	Strong consistency	Strong consistency
Structure	Unstructured	Hierarchically structured	Highly structured at block level
Access Level	Object level	File level	Block level

- Cloud Computing, like any computing, is a combination of CPU, memory, networking, and storage.
- **Infrastructure as a Service** (IaaS) platforms allow you to store your data in either Block Storage or Object Storage formats.
- **Use cases for Block storage**
- **Ideal for databases**, since a DB requires consistent I/O performance and low-latency connectivity.
- Use block storage for **RAID Volumes**, where you combine multiple disks organized through stripping or mirroring.
- Any application which requires **service side processing**, like Java, PHP, and .Net will require block storage.
- Running **mission-critical** applications like Oracle, SAP, Microsoft Exchange, and Microsoft SharePoint.

- Block storage options in the cloud

1. AWS Elastic Block Storage (EBS):

1. Amazon EBS provides raw storage – just like a hard disk – which you can attach to your Elastic Cloud Compute (EC2) instances.
2. Once attached, you create a file system and get immediate access to your storage.
3. You can create EBS General Purpose (SSD) and Provisioned IOPS (SSD) volumes up to 16 TB in size, and slower, legacy magnetic volumes.

2. Rackspace Cloud Block Storage:

1. Rackspace provides raw storage devices capable of delivering super fast 10GbE(Gigabit Ethernet) internal connections.

3. Azure Premium Storage:

1. Premium Storage delivers high-performance, low-latency disk support for I/O intensive workloads running on Azure Virtual Machines. Volumes allow up to 32 TB of storage.

4. Google Persistent Disks:

1. Compute Engine Persistent Disks provide network-attached block storage, much like a high speed and highly reliable SAN, for Compute Engine instances.
2. You can remove a disk from one server and attach it to another server, or attach one volume to multiple nodes in read-only mode.
3. Two types of block storage are available: Standard Persistent Disk and Solid-State Persistent Disks.

- Use Cases for Object Storage
- Storage of **unstructured data** like music, image, and video files.
- Storage for backup files, database dumps, and log files.
- Large data sets. Whether you're storing pharmaceutical or financial data, or multimedia files such as photos and videos, storage can be used as your **big data** object store.
- Archive files in place of local tape drives. Media assets such as **video footage** can be stored in object storage and archived to [AWS glacier](#).

- Object storage options in the Cloud

1. Amazon S3:

1. [Amazon S3](#) stores data as objects within resources called “buckets.”
2. AWS S3 offers features like 99.999999999% durability, cross-region replication, event notifications, versioning, encryption, and flexible storage options (redundant and standard).

2. Rackspace Cloud Files:

1. Cloud Files provides online object storage for files and media.
2. Cloud Files writes each file to three storage disks on separate nodes that have dual power supplies.
3. All traffic between your application and Cloud Files uses SSL to establish a secure, encrypted channel.
4. You can host static websites (for example: blogs, brochure sites, small company sites) entirely from Cloud Files with a global CDN.

- Object storage options in the Cloud

3. Azure Blob Storage:

1. For users with large amounts of unstructured data to store in the cloud, [Blob storage](#) offers a cost-effective and scalable solution.
2. Every blob is organized into a container with up to a 500 TB storage account capacity limit.

4. Google cloud storage:

1. Cloud Storage allows you to store data in Google's cloud.
2. [Google Cloud Storage](#) supports individual objects that are terabytes in size.
3. It also supports a large number of buckets per account.
4. Google Cloud Storage provides strong read-after-write consistency for all upload and delete operations.
5. Two types of storage class are available: Standard Storage class and Storage Near line class (with Near Line being **MUCH** cheaper).

- **What is Shell?**
- **Shell** is a UNIX term for an interface between a user and an operating system service.
- Shell provides users with an interface and accepts human-readable commands into the system and executes those commands which can run automatically and give the program's output in a shell script.
- An Operating is made of many components, but its two prime components are –
 - Kernel
 - Shell



- **What is Shell?**
- A Kernel is at the nucleus of a computer.
- It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one.
- A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it.
- When you run the terminal, the Shell issues **a command prompt (usually \$)**, where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal.
- The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name **Shell**.
- This Unix/Linux Shell Script tutorial helps understand shell scripting basics to advanced levels.

- **What is Shell?**
- A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output.
- It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it.
- When you run the terminal, the Shell issues **a command prompt (usually \$)**, where you can type your input, which is then executed when you hit the Enter key.
- The output or the result is thereafter displayed on the terminal.
- The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name **Shell**.
- This Unix/Linux Shell Script tutorial helps understand shell scripting basics to advanced levels.

- **Shell Scripting**
- **Shell Scripting** is an open-source computer program designed to be run by the Unix/Linux shell.
- Shell Scripting is a program to write a series of commands for the shell to execute.
- It can combine lengthy and repetitive sequences of commands into a single and simple script that can be stored and executed anytime which, reduces programming efforts.

- **Types of Shell**

- There are two main shells in Linux:

1. The Bourne Shell: The prompt for this shell is \$ and its derivatives are listed below:

- POSIX shell also is known as sh
- Korn Shell also knew as sh
- Bourne Again SHell also knew as bash (most popular)

2. The C shell: The prompt for this shell is %, and its subcategories are:

- C shell also is known as csh
- Tops C shell also is known as tcsh

- **How to Write Shell Script in Linux/Unix**
- **Shell Scripts** are written using text editors.
- On your Linux system, open a text editor program, open a new file to begin typing a shell script or shell programming, then give the shell permission to execute your shell script and put your script at the location from where the shell can find it.
- Let us understand the steps in creating a Shell Script:
 - 1.Create a file using a vi editor(or any other editor). Name script file with extension .sh**
 - 2.Start the script with #! /bin/sh**
 - 3.Write some code.**
 - 4.Save the script file as filename.sh**
 - 5.For executing the script type bash filename.sh**

- “#!” is an operator called shebang which directs the script to the interpreter location.
- So, if we use ”#! /bin/sh” the script gets directed to the bourne-shell.
- Let’s create a small script –
- `#!/bin/sh`
- `ls`



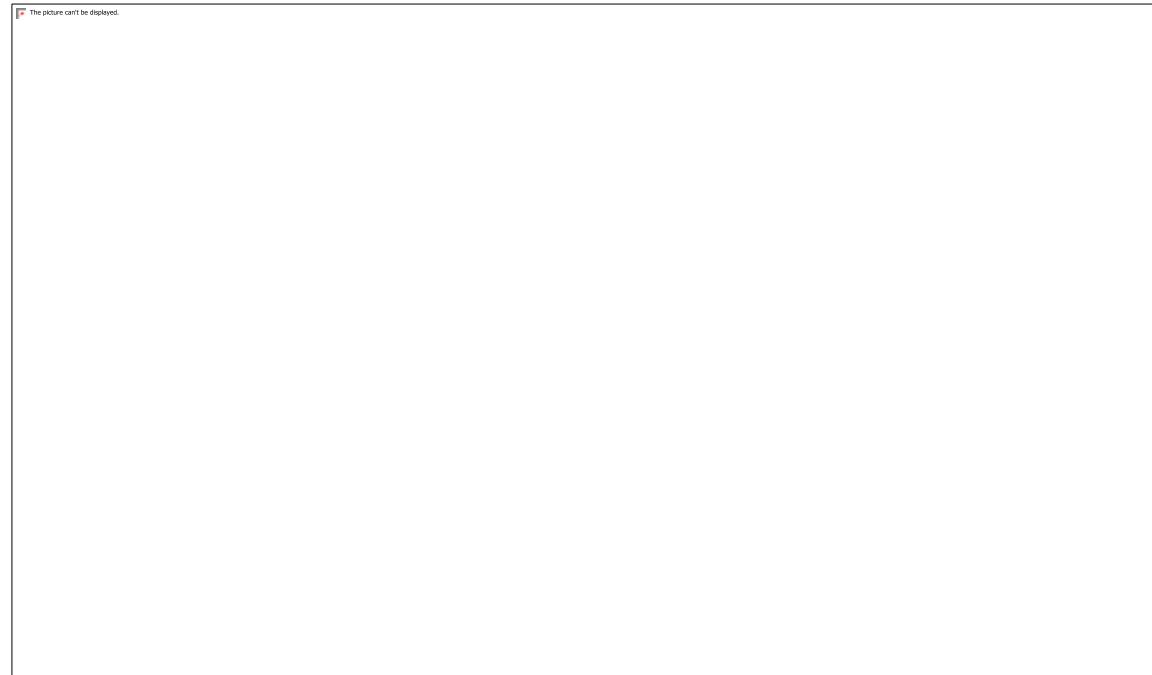
- Steps to Create Shell Script in Linux/Unix

Command ‘ls’ is executed when we execute the script sample.sh file.



Steps to Create Shell Script in Linux/Unix

- **Adding shell comments**
- Commenting is important in any program. In Shell programming, the syntax to add a comment is
- `#comment`
- Let understand this with an example.



- **echo** command in Linux is used to display line of text/string that are passed as an argument . This is a built in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.
- echo [option] [string]
- **Displaying a text/string : Syntax :**
- echo [string]

- Defining Variables
- Variables are defined as follows –
- variable_name=variable_value
- For example –
- NAME=“VIIT”
- The above example defines the variable NAME and assigns the value “VIIT" to it.
- Variables of this type are called **scalar variables**.
- A scalar variable can hold only one value at a time.
- Shell enables you to store any value you want in a variable. For example –
- VAR1=“VIIT”
- VAR2=100

- Accessing Values
- To access the value stored in a variable, prefix its name with the dollar sign (\$) –
- For example, the following script will access the value of defined variable NAME and print it on

- `#!/bin/sh`

- `NAME="VIIT"`
- `echo $NAME`
- The above script will produce the following value –
- `VIIT`

- Shell Scripting Variables
- Scripts can contain variables inside the script.

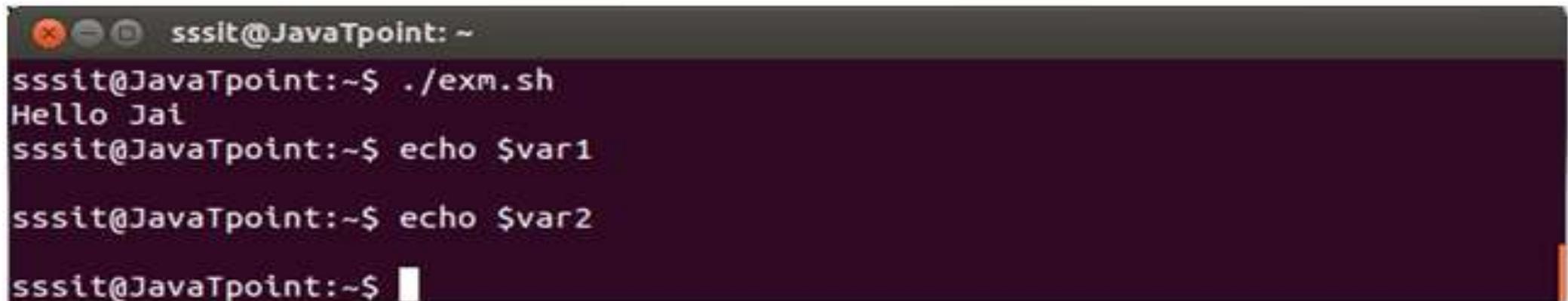


```
sssit@JavaPoint: ~
#!/bin/bash

var1=Hello
var2=Jai
#
echo "$var1 $var2"
```

- Look at the above snapshot, two variables are assigned to the script **\$var1** and **\$var2**.
- As scripts run in their own shell, hence variables do not survive the end of the script.

- Look at the above snapshot, two variables are assigned to the script \$var1 and \$var2.
- As scripts run in their own shell, hence variables do not survive the end of the script.]



```
sssit@JavaPoint: ~
sssit@JavaPoint:~$ ./exm.sh
Hello Jai
sssit@JavaPoint:~$ echo $var1
sssit@JavaPoint:~$ echo $var2
sssit@JavaPoint:~$ █
```

A screenshot of a terminal window titled "sssit@JavaPoint: ~". The user runs the command "./exm.sh", which outputs "Hello Jai". Then, the user runs "echo \$var1" and "echo \$var2", both of which return empty results. The terminal has a dark background with light-colored text and standard OS X-style window controls.

- Look at the above snapshot, **var1** and **var2** do not run outside the script.

- **Read-only Variables**
- Shell provides a way to mark variables as read-only by using the readonly command. After a variable is marked read-only, its value cannot be changed.
- For example, the following script generates an error while trying to change the value of NAME –
- `#!/bin/sh`

- `NAME="VIIT"`
- `readonly NAME`
- `NAME="CSE"`
- The above script will generate the following result –
- `/bin/sh: NAME: This variable is read only.`

- **Variable Types**
- When a shell is running, three main types of variables are present –
- **Local Variables** – A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at the command prompt.
- **Environment Variables** – An environment variable is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually, a shell script defines only those environment variables that are needed by the programs that it runs.
- **Shell Variables** – A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

- Special Variables
- For example, the \$ character represents the process ID number, or PID, of the current shell –
- \$echo \$\$
- The above command writes the PID of the current shell –
- 29949

- Shell supports a different type of variable called an **array variable**.
 - This can hold multiple values at the same time.
 - Arrays provide a method of grouping a set of variables.
 - Instead of creating a new name for each variable that is required, you can use a single array variable that stores all the other variables.
 - All the naming rules discussed for Shell Variables would be applicable while naming arrays.
 - **Defining Array Values**
 - The difference between an array variable and a scalar variable can be explained as follows.
 - Suppose you are trying to represent the names of various students as a set of variables. Each of the individual variables is a scalar variable as follows
-

- NAME01="Zara"
- NAME02="Qadir"
- NAME03="Mahnaz"
- NAME04="Ayan"
- NAME05="Daisy"
- We can use a single array to store all the above mentioned names.
- Following is the simplest method of creating an array variable.
- This helps assign a value to one of its indices.
- **array_name[index]=value**

- Here *array_name* is the name of the array, *index* is the index of the item in the array that you want to set, and *value* is the value you want to set for that item.
- As an example, the following commands –
- NAME[0] = "Zara"
- NAME[1] = "Qadir"
- NAME[2] = "Mahnaz"
- NAME[3] = "Ayan"
- NAME[4] = "Daisy"
- If you are using the **ksh** shell, here is the syntax of array initialization –
- `set -A array_name value1 value2 ... valuen`

- If you are using the **bash** shell, here is the syntax of array initialization –
- `array_name=(value 1 ... Value n)`
- Accessing Array Values
- After you have set any array variable, you access it as follows –
- `${array_name[index]}`
- Here *array_name* is the name of the array, and *index* is the index of the value to be accessed. Following is an example to understand the concept –

- `#!/bin/sh`
- `NAME[0] = "Zara"`
- `NAME[1] = "Qadir"`
- `NAME[2] = "Mahnaz"`
- `NAME[3] = "Ayan"`
- `NAME[4] = "Daisy"`
- `echo "First Index: ${NAME[0]}"`
- `echo "Second Index: ${NAME[1]}"`

- The above example will generate the following result –
- `./test.sh`
- First Index: Zara
- Second Index: Qadir

- You can access all the items in an array in one of the following ways –
- \${array_name[*]}
- \${array_name[@]}
- Here **array_name** is the name of the array you are interested in. Following example will help you understand the concept –

- #!/bin/sh
- NAME[0] = "VIIT"
- NAME[1] = "COMP"
- NAME[2] = "ENTC"
- NAME[3] = "CIVIL"
- NAME[4] = "IT"
- echo "First Method: \${NAME[*]}"
- echo "Second Method: \${NAME[@]}"

References

- 1) <https://www.redhat.com/en/topics/data-storage/file-block-object-storage>
- 2) <https://www.guru99.com/introduction-to-shell-scripting.html>
- 3) <https://www.geeksforgeeks.org/echo-command-in-linux-with-examples/>
- 4) https://www.tutorialspoint.com/execute_bash_online.php
- 5) <https://www.tutorialspoint.com/unix/unix-using-arrays.htm>

Firewalls

Presented By
Santosh Kumar

Outline

- Introduction
- Firewall Environments
- Type of Firewalls
- Future of Firewalls
- Conclusion

Introduction

- Firewalls control the flow of network traffic
- Firewalls have applicability in networks where there is no internet connectivity
- Firewalls operate on number of layers
- Can also act as VPN gateways
- Active content filtering technologies

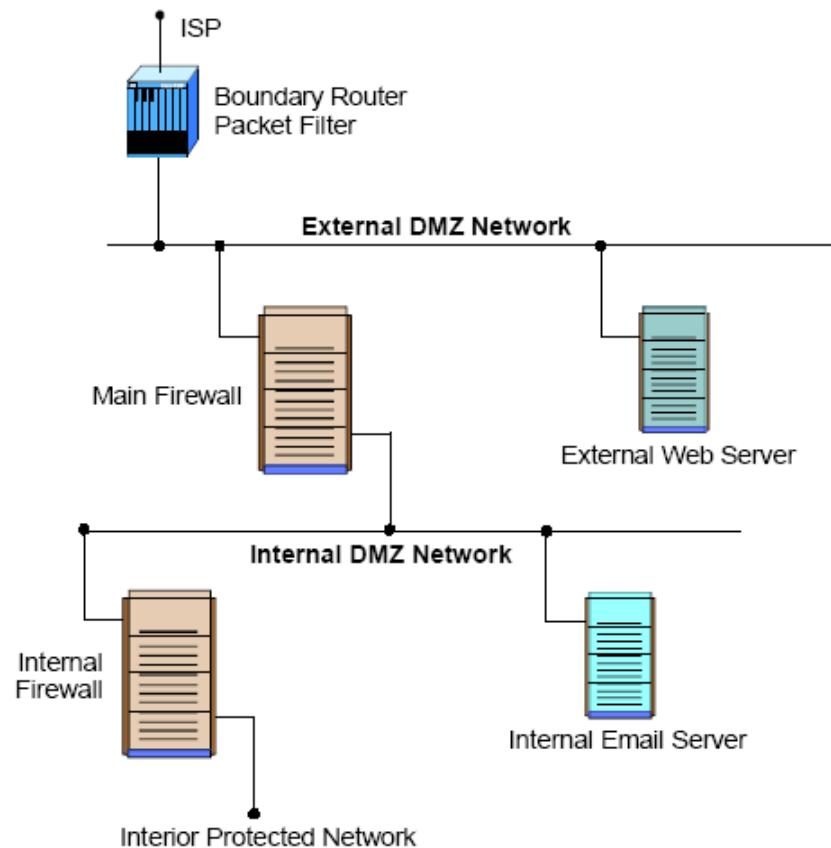
Firewall Environments

- There are different types of environments where a firewall can be implemented.
- Simple environment can be a packet filter firewall
- Complex environments can be several firewalls and proxies

DMZ Environment

- Can be created out of a network connecting two firewalls
- Boundary router filter packets protecting server
- First firewall provide access control and protection from server if they are hacked

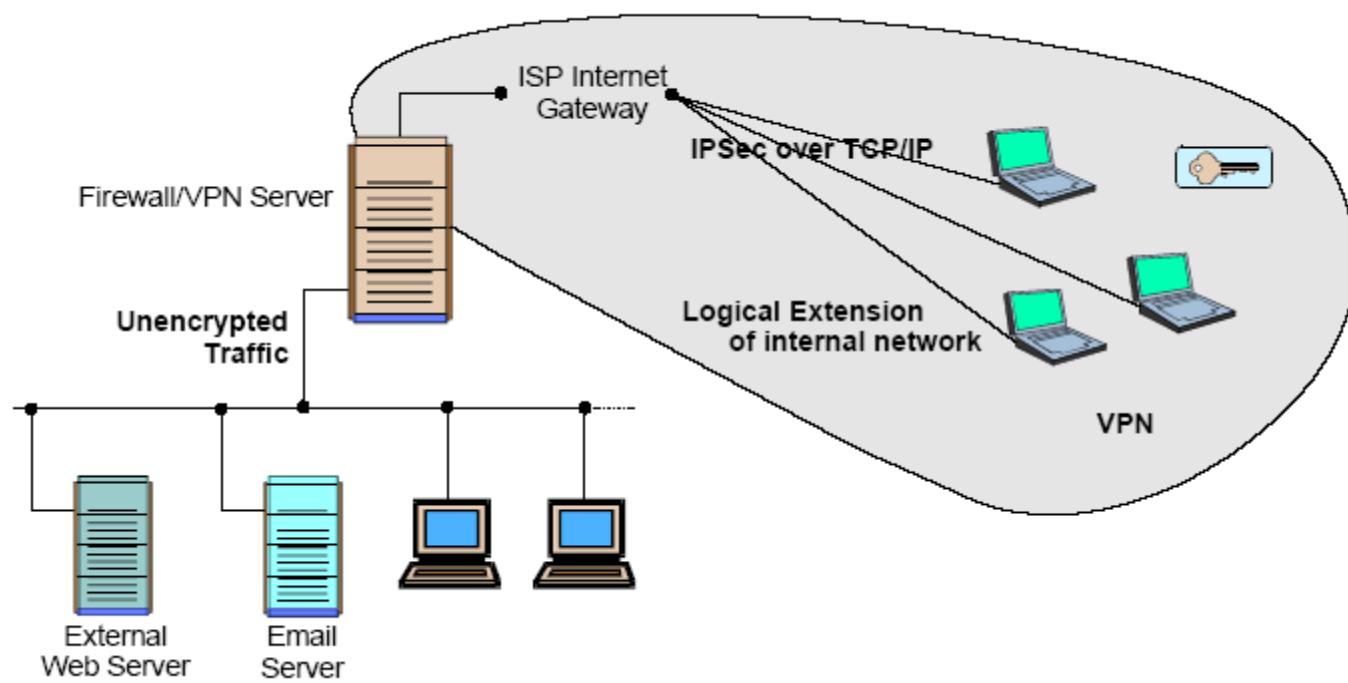
DMZ ENV



VPN

- VPN is used to provide secure network links across networks
- VPN is constructed on top of existing network media and protocols
- On protocol level IPsec is the first choice
- Other protocols are PPTP, L2TP

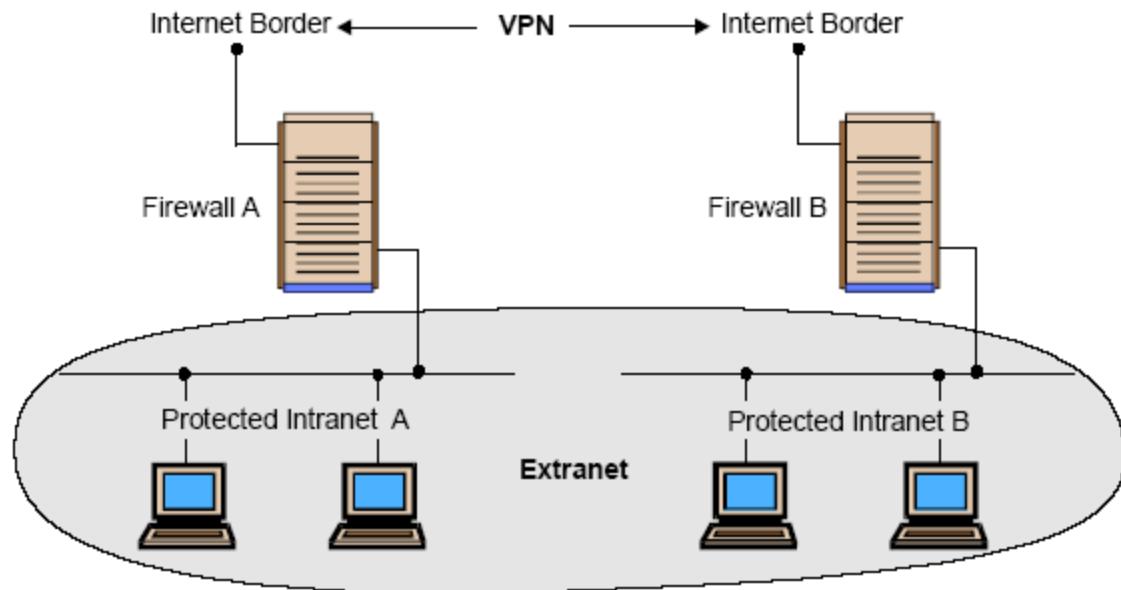
VPN



Intranets

- An intranet is a network that employs the same types of services, applications, and protocols present in an Internet implementation, without involving external connectivity
- Intranets are typically implemented behind firewall environments.

Intranets



Extranets

- Extranet is usually a business-to-business intranet
- Controlled access to remote users via some form of authentication and encryption such as provided by a VPN
- Extranets employ TCP/IP protocols, along with the same standard applications and services

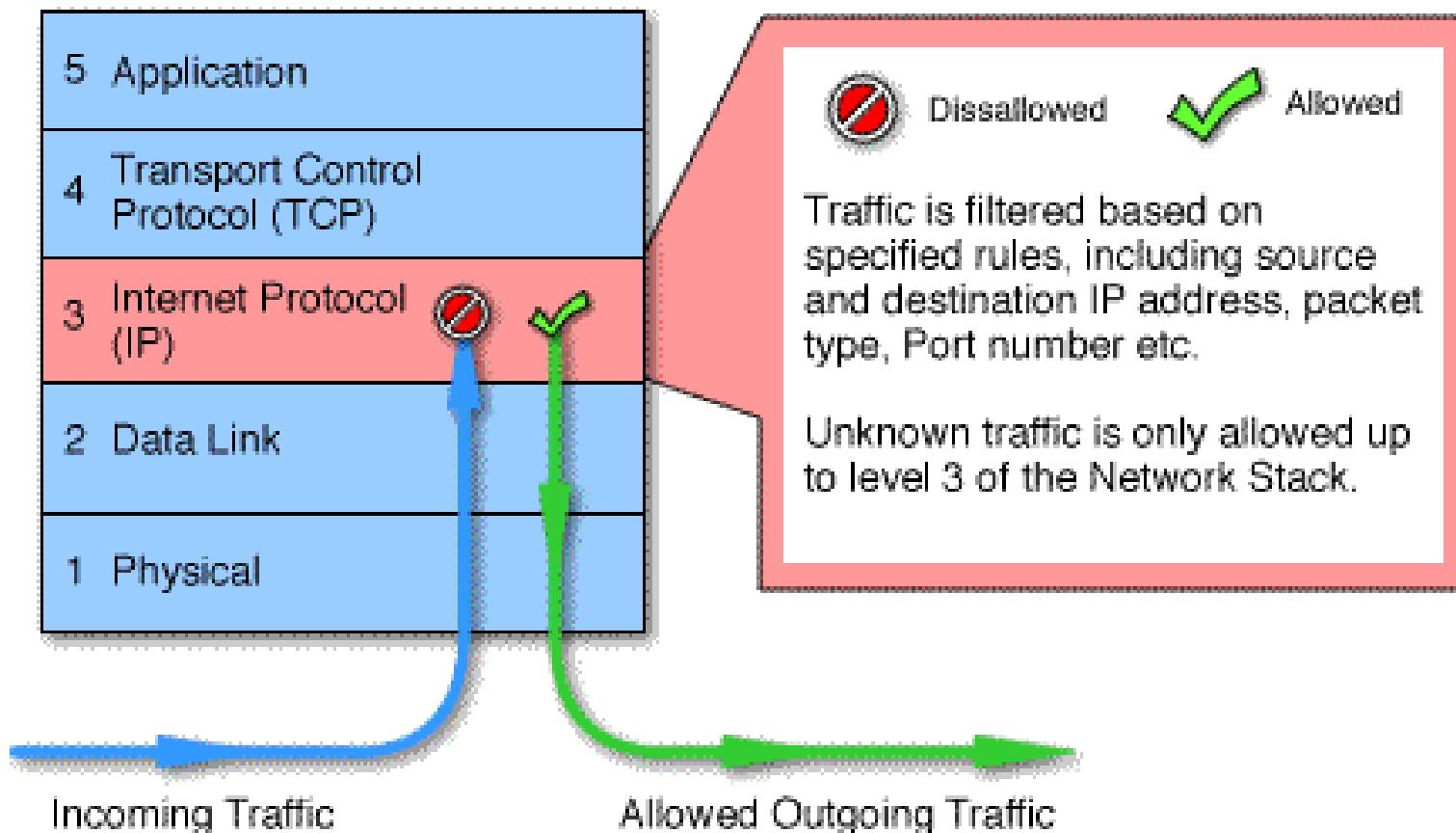
Type is Firewalls

- Firewalls fall into four broad categories
- Packet filters
- Circuit level
- Application level
- Stateful multilayer

Packet Filter

- Work at the network level of the OSI model
- Each packet is compared to a set of criteria before it is forwarded
- Packet filtering firewalls is low cost and low impact on network performance

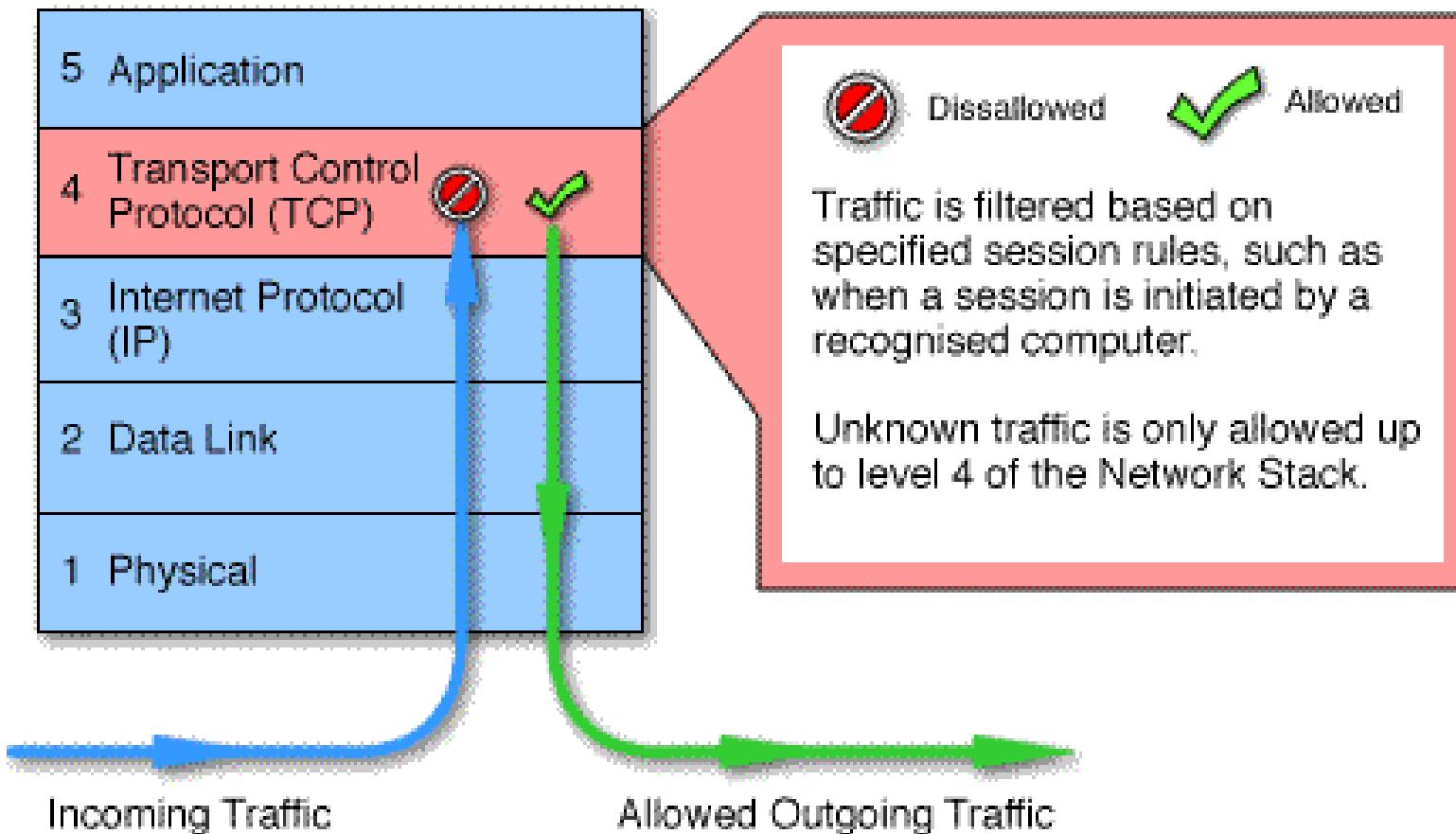
Packet Filtering



Circuit level

- Circuit level gateways work at the session layer of the OSI model, or the TCP layer of TCP/IP
- Monitor TCP handshaking between packets to determine whether a requested session is legitimate.

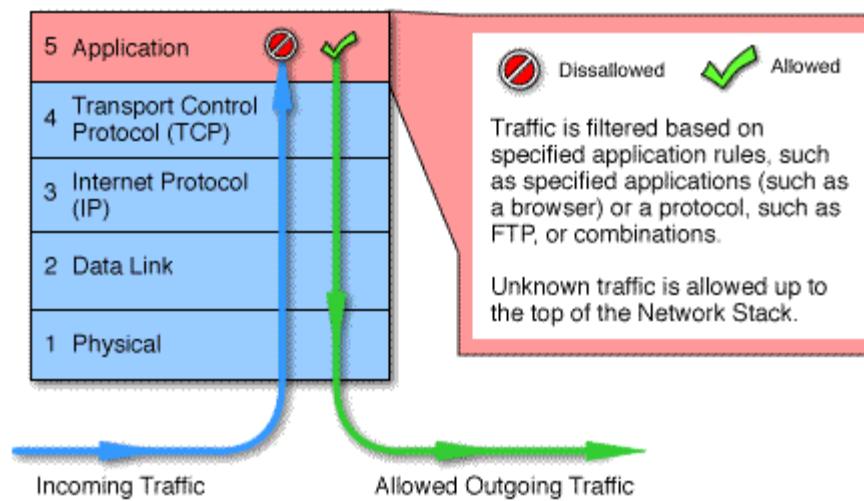
Circuit Level



Application Level

- Application level gateways, also called proxies, are similar to circuit-level gateways except that they are application specific
- Gateway that is configured to be a web proxy will not allow any ftp, gopher, telnet or other traffic through

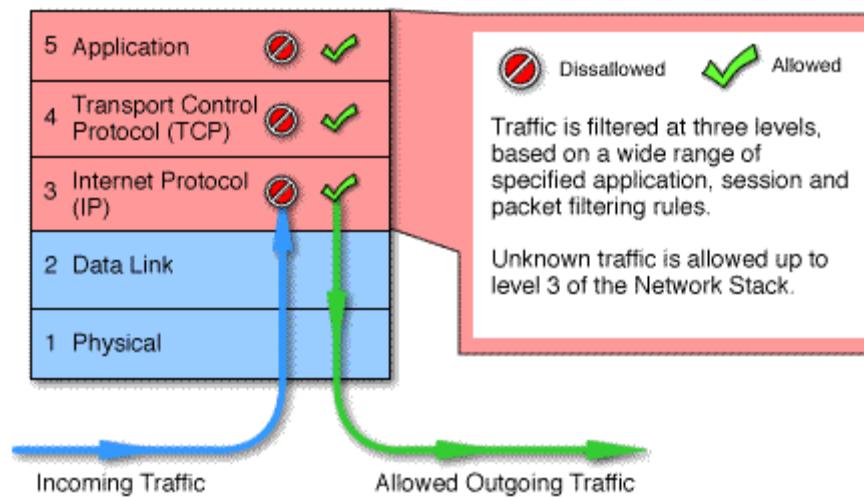
Application Level



Stateful Multilayer

- Stateful multilayer inspection firewalls combine the aspects of the other three types of firewalls
- They filter packets at the network layer, determine whether session packets are legitimate and evaluate contents of packets at the application layer

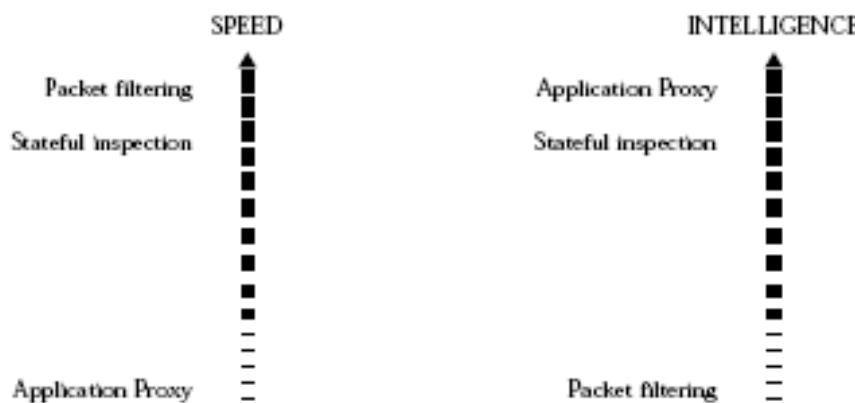
Stateful Multilayer



General Performance

FIREWALL PERFORMANCE SUMMARY

Technology	Speed	Flexibility	Intelligence
Packet filtering	V. Good	V.Good	Low
Application Proxy	Low	Low	V. Good
Stateful inspection	Good	Good	Good
Circuit gateway	Low	Low	Low



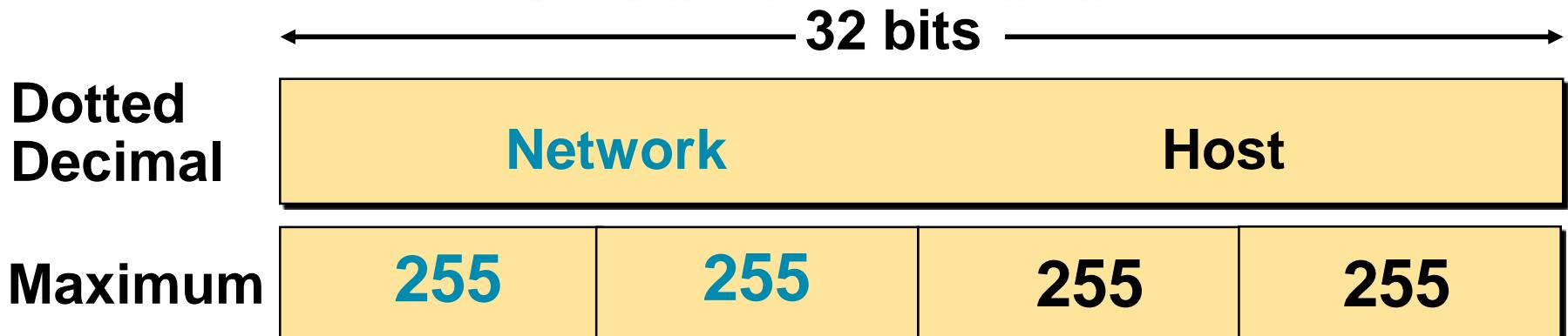
Future of Firewalls

- Firewalls will continue to advance as the attacks on IT infrastructure become more and more sophisticated
- More and more client and server applications are coming with native support for proxied environments
- Firewalls that scan for viruses as they enter the network and several firms are currently exploring this idea, but it is not yet in wide use

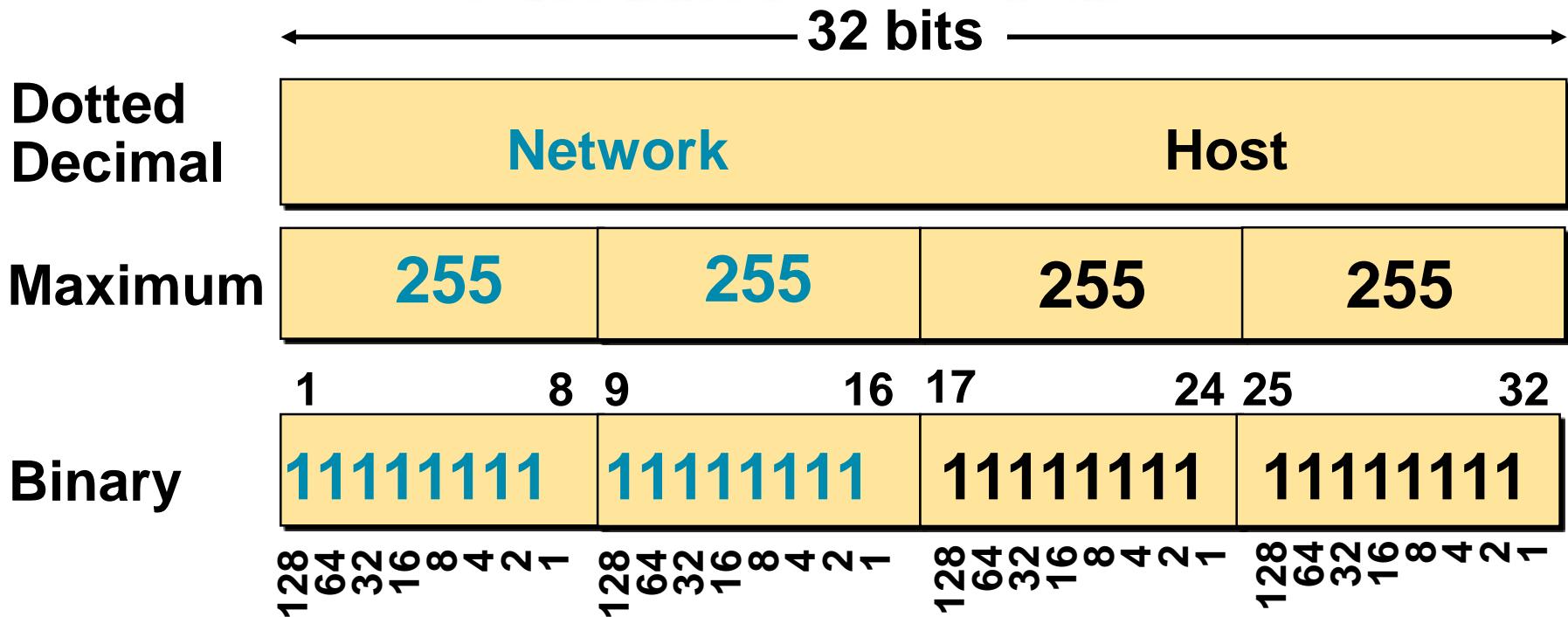
Conclusion

- It is clear that some form of security for private networks connected to the Internet is essential
- A firewall is an important and necessary part of that security, but cannot be expected to perform all the required security functions.

IP Addressing



IP Addressing



IP Addressing

32 bits			
Dotted Decimal	Network		Host
Maximum	255	255	255
	1	8 9	16 17
Binary	11111111	11111111	11111111
	128 64 32 16 8 4 2 1	128 64 32 16 8 4 2 1	128 64 32 16 8 4 2 1
Example Decimal	172	16	122
Example Binary	10101100	00010000	01111010
	11001100		

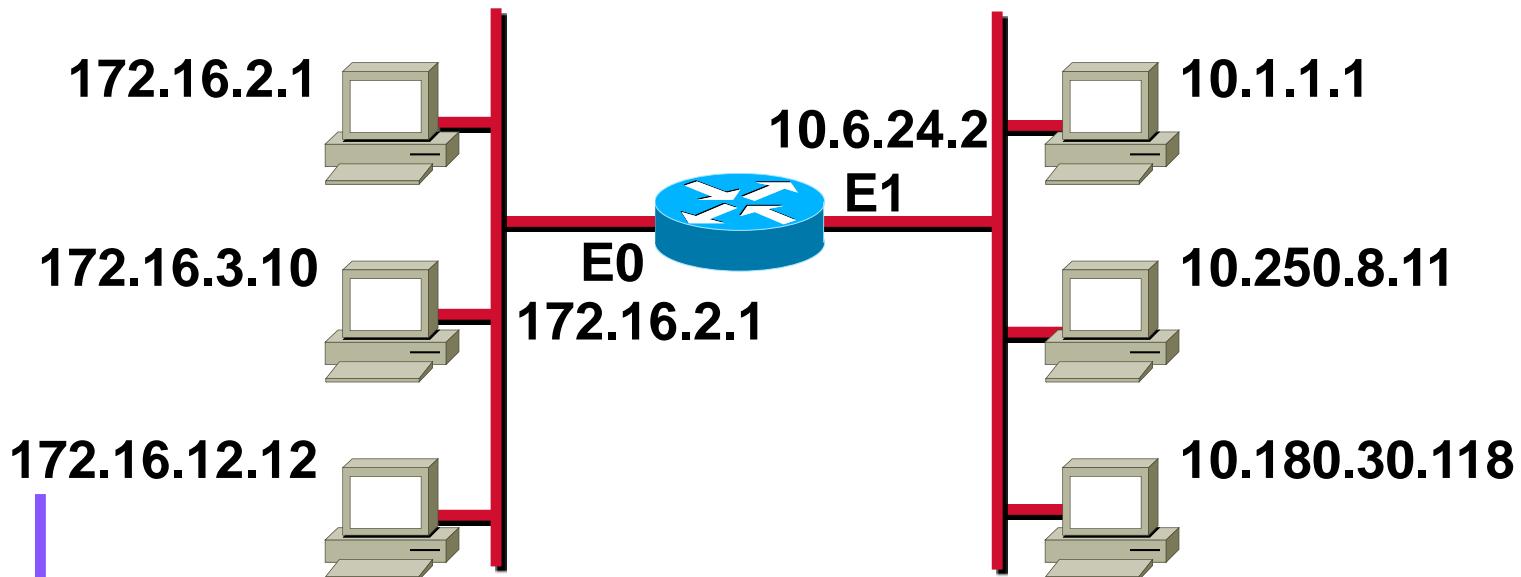
IP Address Classes

	8 bits	8 bits	8 bits	8 bits
Class A:	Network	Host	Host	Host
Class B:	Network	Network	Host	Host
Class C:	Network	Network	Network	Host
Class D:	Multicast			
Class E:	Research			

IP Address Classes

Bits:	1	8 9	16 17	24 25	32
Class A:	0NNNNNNN	Host	Host	Host	
	Range (1-126)				
Class B:	10NNNNNN	Network	Host	Host	
	Range (128-191)				
Class C:	110NNNNN	Network	Network	Host	
	Range (192-223)				
Class D:	1110MMMM	Multicast Group	Multicast Group	Multicast Group	
	Range (224-239)				

Host Addresses



172.16 . 12 . 12
Network Host

Routing Table	
Network	Interface
172.16.0.0	E0
10.0.0.0	E1

Determining Available Host Addresses

Network		Host		
172	16	0	0	
10101100	00010000	00000000	00000000	
		00000000	00000001	1
		00000000	00000011	2
				3
				⋮
		11111111	11111101	65534
		11111111	11111110	65535
		11111111	11111111	65536
			- 2	⋮
$2^N - 2 = 2^{16} - 2 = 65534$			$\frac{65534}{65534}$	

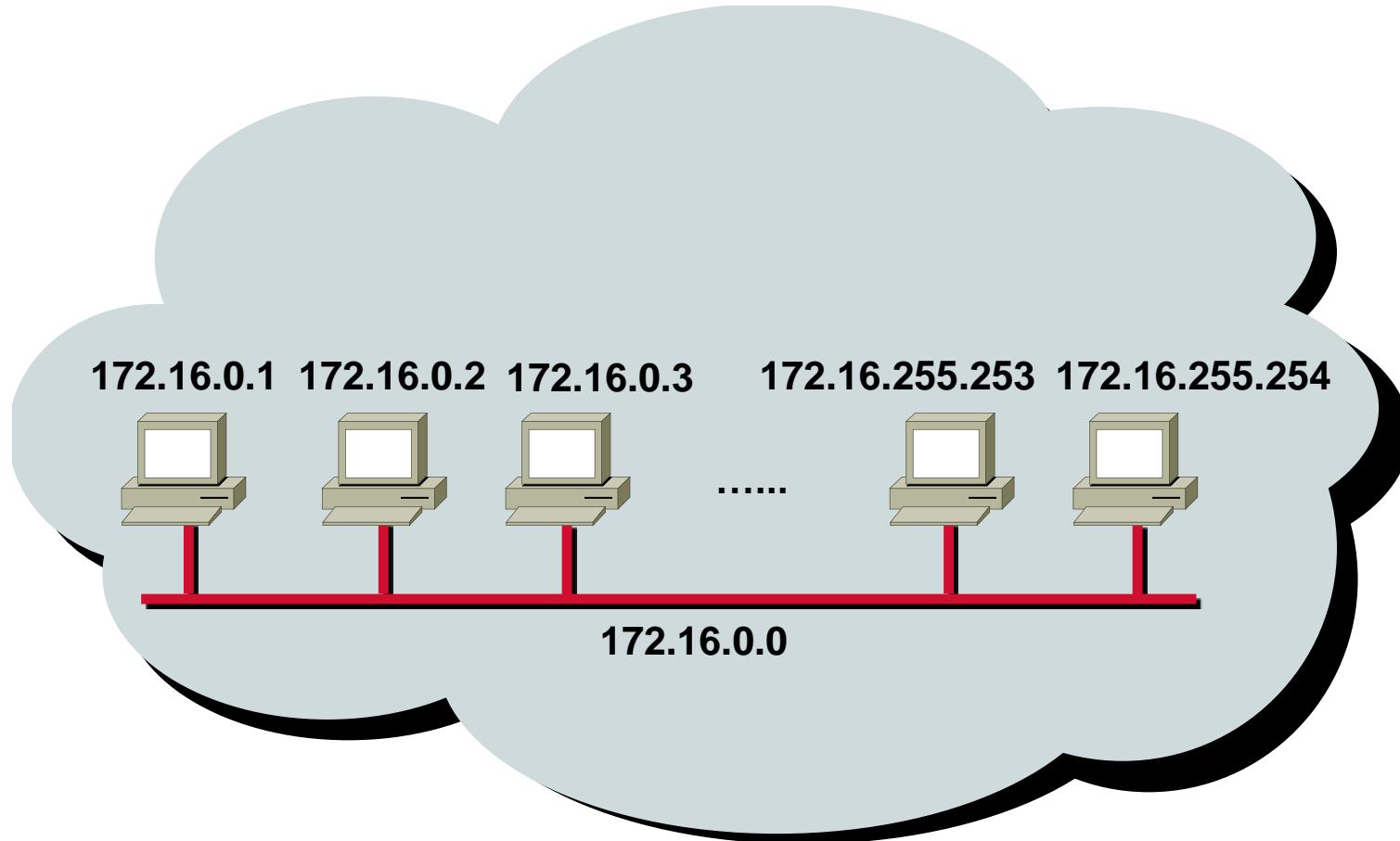
IP Address Classes Exercise

Address	Class	Network	Host
10.2.1.1			
128.63.2.100			
201.222.5.64			
192.6.141.2			
130.113.64.16			
256.241.201.10			

IP Address Classes Exercise Answers

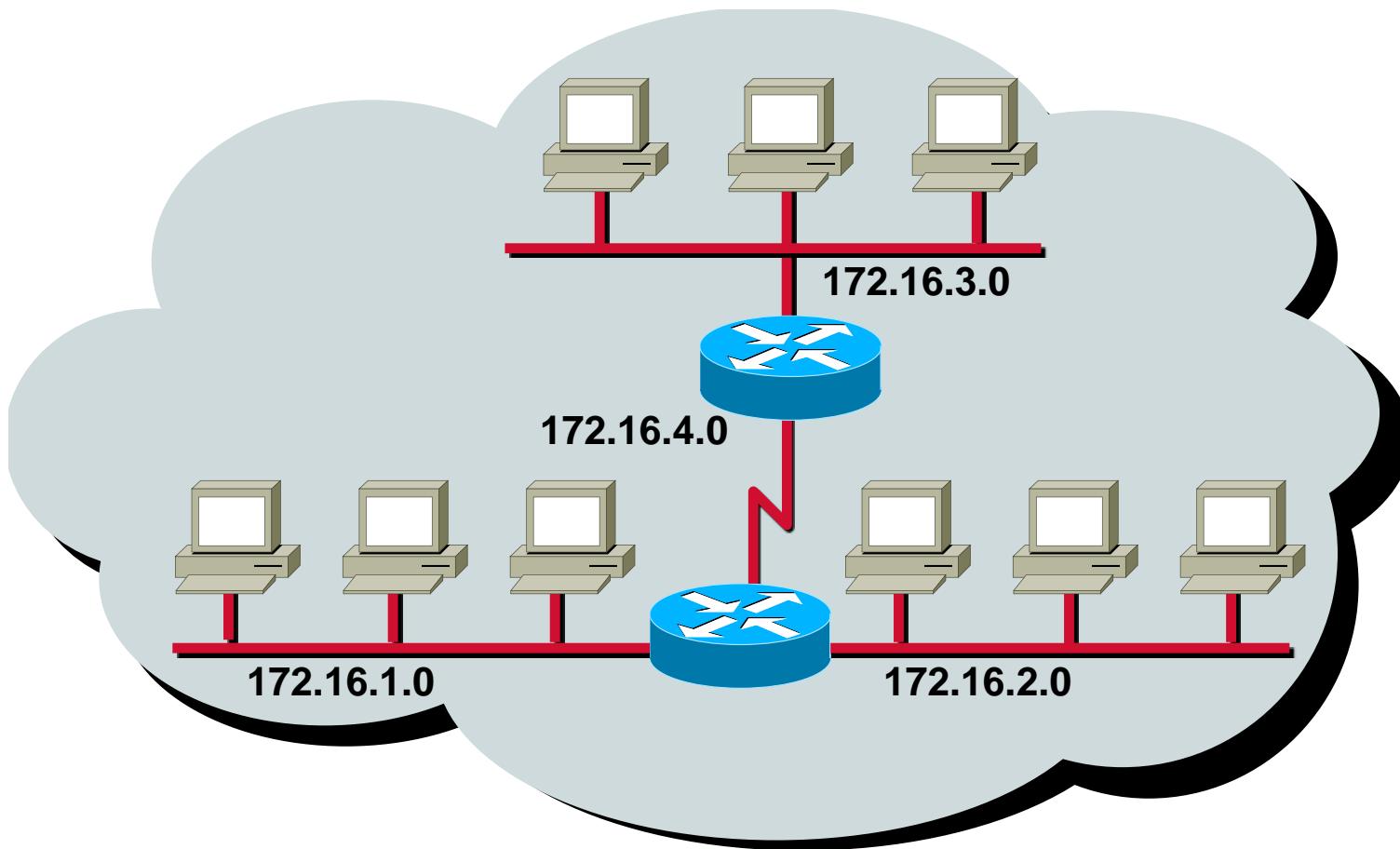
Address	Class	Network	Host
10.2.1.1	A	10.0.0.0	0.2.1.1
128.63.2.100	B	128.63.0.0	0.0.2.100
201.222.5.64	C	201.222.5.0	0.0.0.64
192.6.141.2	C	192.6.141.0	0.0.0.2
130.113.64.16	B	130.113.0.0	0.0.64.16
256.241.201.10	Nonexistent		

Addressing without Subnets



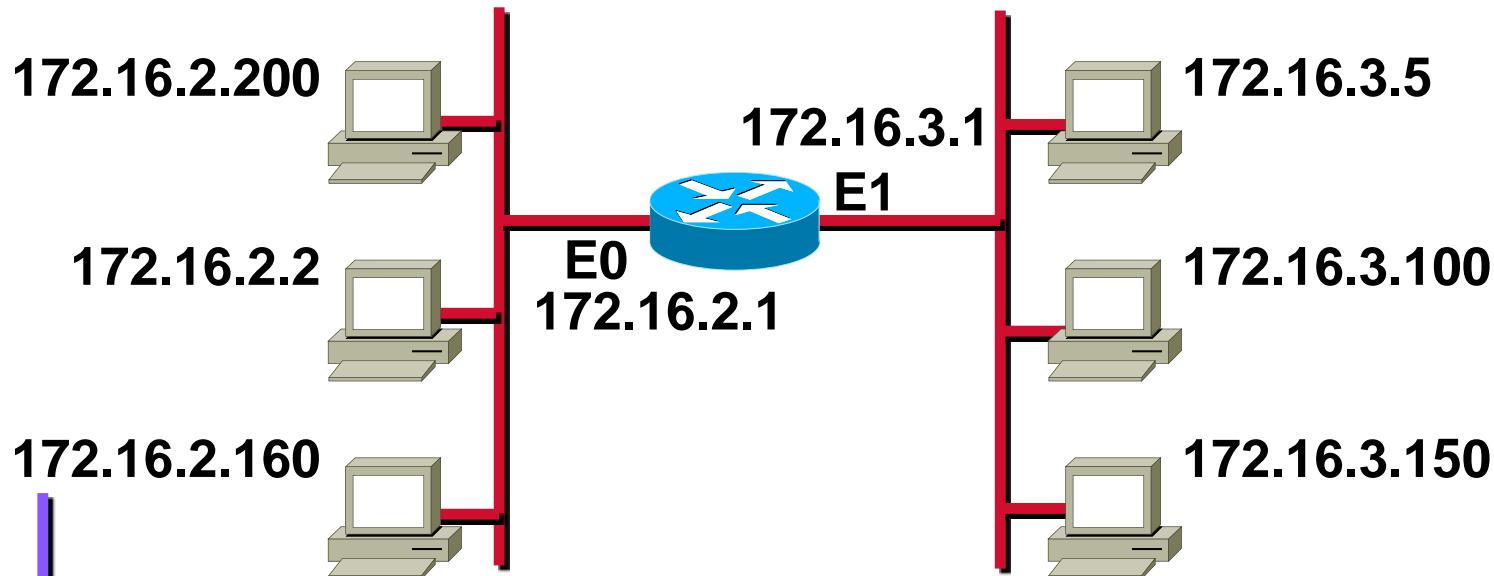
Network 172.16.0.0

Addressing with Subnets



Network 172.16.0.0

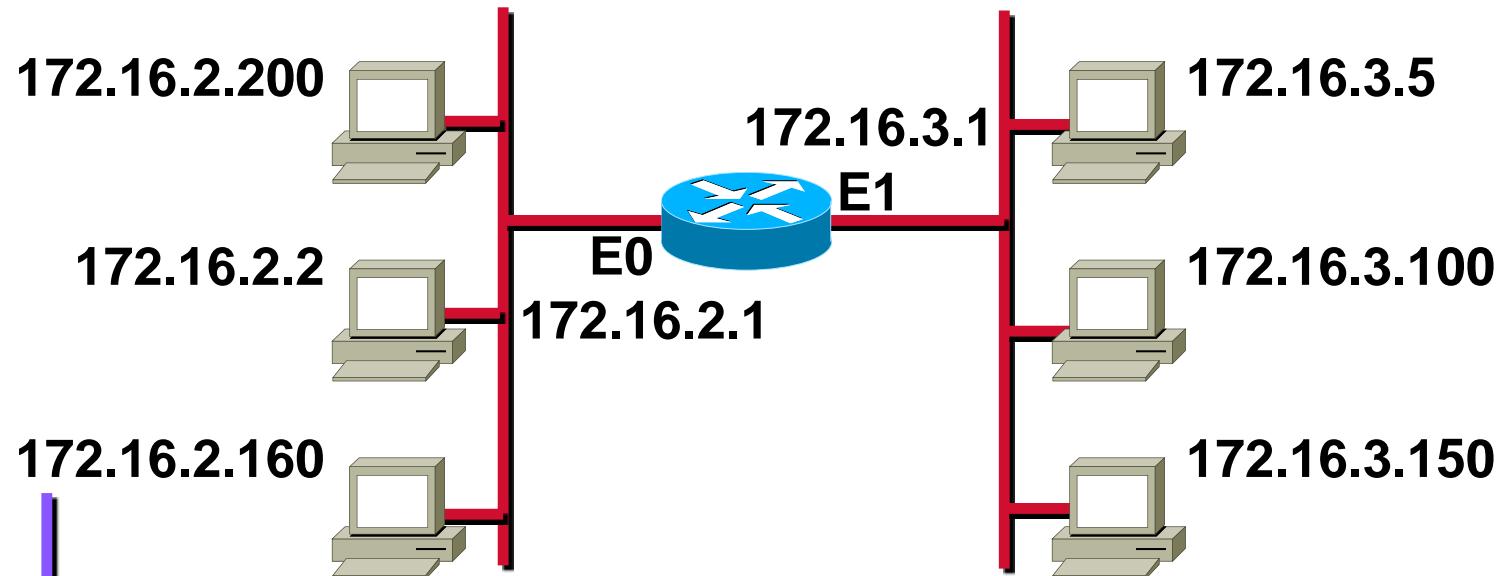
Subnet Addressing



172.16 . 2 . 160
Network Host

Network	Interface
172.16.0.0	E0
172.16.0.0	E1

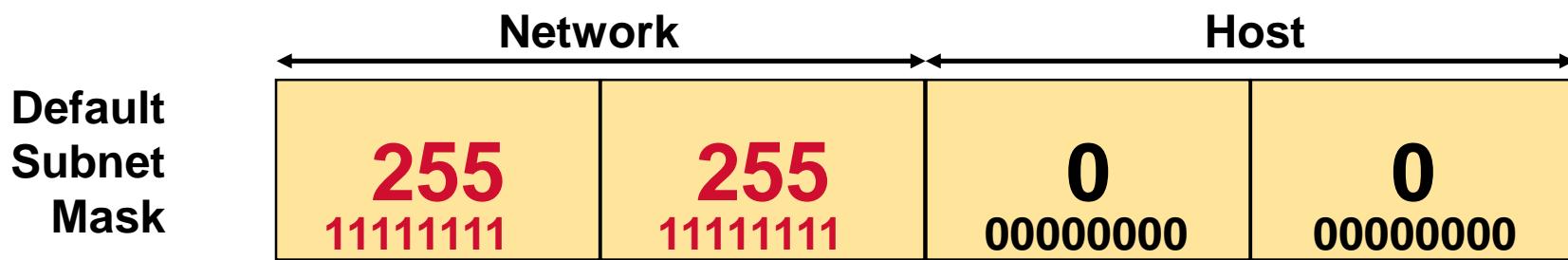
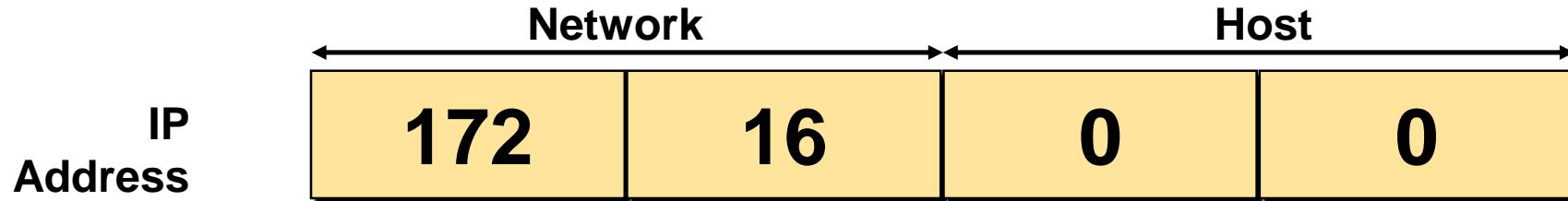
Subnet Addressing



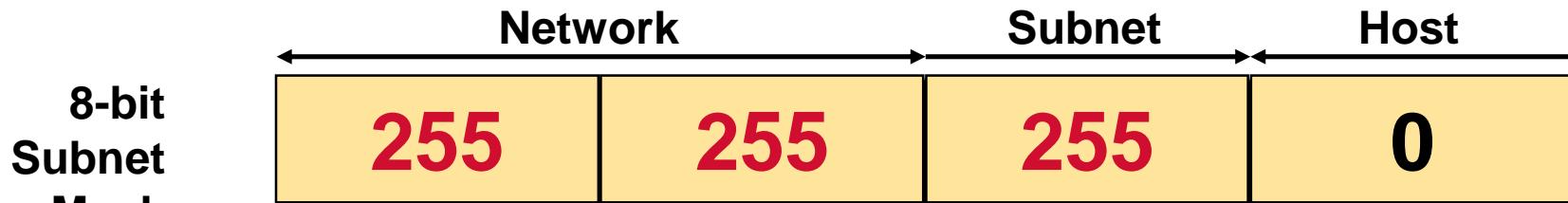
172.16 . 2 . 160
Network Subnet Host

Network	Interface
172.16.2.0	E0
172.16.3.0	E1

Subnet Mask



Also written as “/16” where 16 represents the number of 1s in the mask.



Also written as “/24” where 24 represents the number of 1s in the mask.

Decimal Equivalents of Bit Patterns

128	64	32	16	8	4	2	1	=	
1	0	0	0	0	0	0	0	=	128
1	1	0	0	0	0	0	0	=	192
1	1	1	0	0	0	0	0	=	224
1	1	1	1	0	0	0	0	=	240
1	1	1	1	1	0	0	0	=	248
1	1	1	1	1	1	0	0	=	252
1	1	1	1	1	1	1	0	=	254
1	1	1	1	1	1	1	1	=	255

Subnet Mask without Subnets

	Network		Host	
172.16.2.160	10101100	00010000	00000010	10100000
255.255.0.0	11111111	11111111	00000000	00000000
	10101100	00010000	00000000	00000000
Network Number	172	16	0	0

Subnets not in use—the default

Subnet Mask with Subnets

	Network	Subnet	Host	
172.16.2.160	10101100	00010000	00000010	10100000
255.255.255.0	11111111	11111111	11111111	00000000
	10101100	00010000	00000010	00000000

128
192
224
240
248
252
254
255

Network Number

172	16	2	0
-----	----	---	---

Network number extended by eight bits

Subnet Mask with Subnets (cont.)

	Network	Subnet	Host	
172.16.2.160	10101100	00010000	00000010	10100000
255.255.255.192	11111111	11111111	11111111	11000000
	10101100	00010000	00000010	10000000

128 192 224 240 248 252 254 255

128 192 224 240 248 252 254 255

Network Number

172	16	2	128
-----	----	---	-----

Network number extended by ten bits

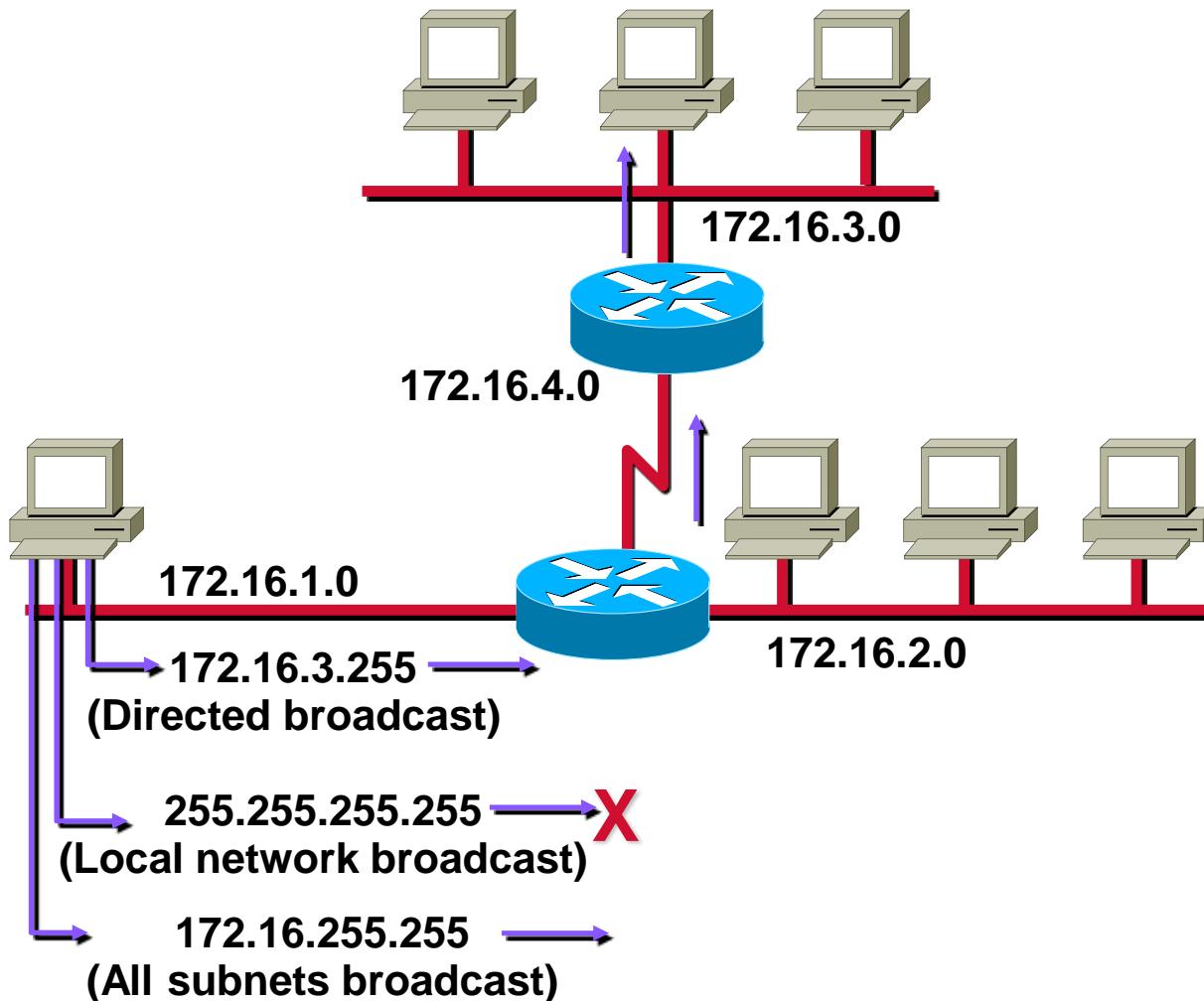
Subnet Mask Exercise

Address	Subnet Mask	Class	Subnet
172.16.2.10	255.255.255.0		
10.6.24.20	255.255.240.0		
10.30.36.12	255.255.255.0		

Subnet Mask Exercise Answers

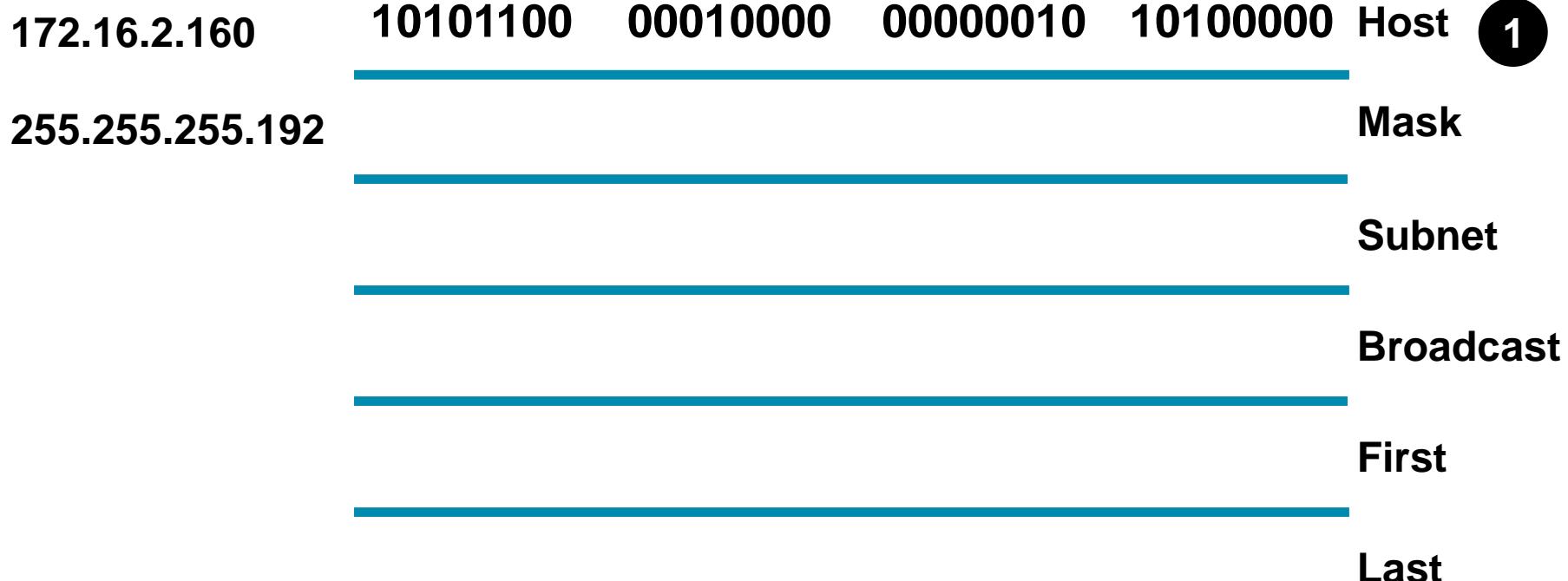
Address	Subnet Mask	Class	Subnet
172.16.2.10	255.255.255.0	B	172.16.2.0
10.6.24.20	255.255.240.0	A	10.6.16.0
10.30.36.12	255.255.255.0	A	10.30.36.0

Broadcast Addresses



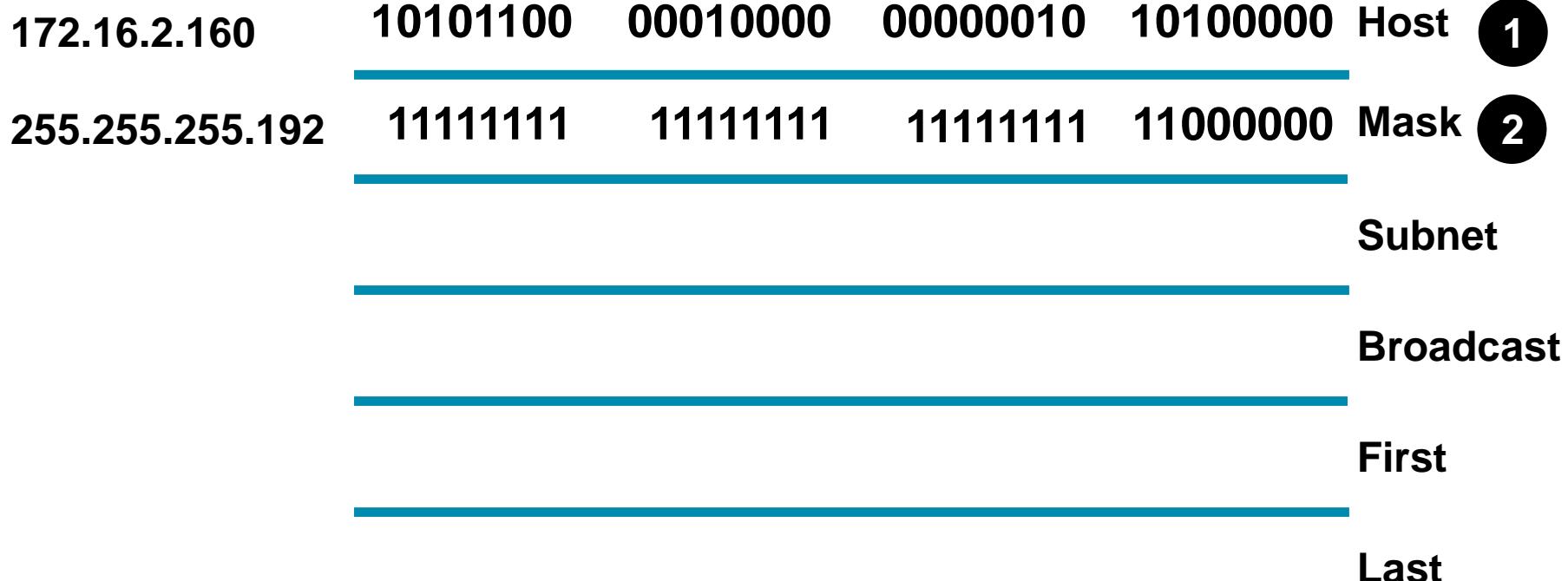
Addressing Summary Example

172	16	2	160
-----	----	---	-----

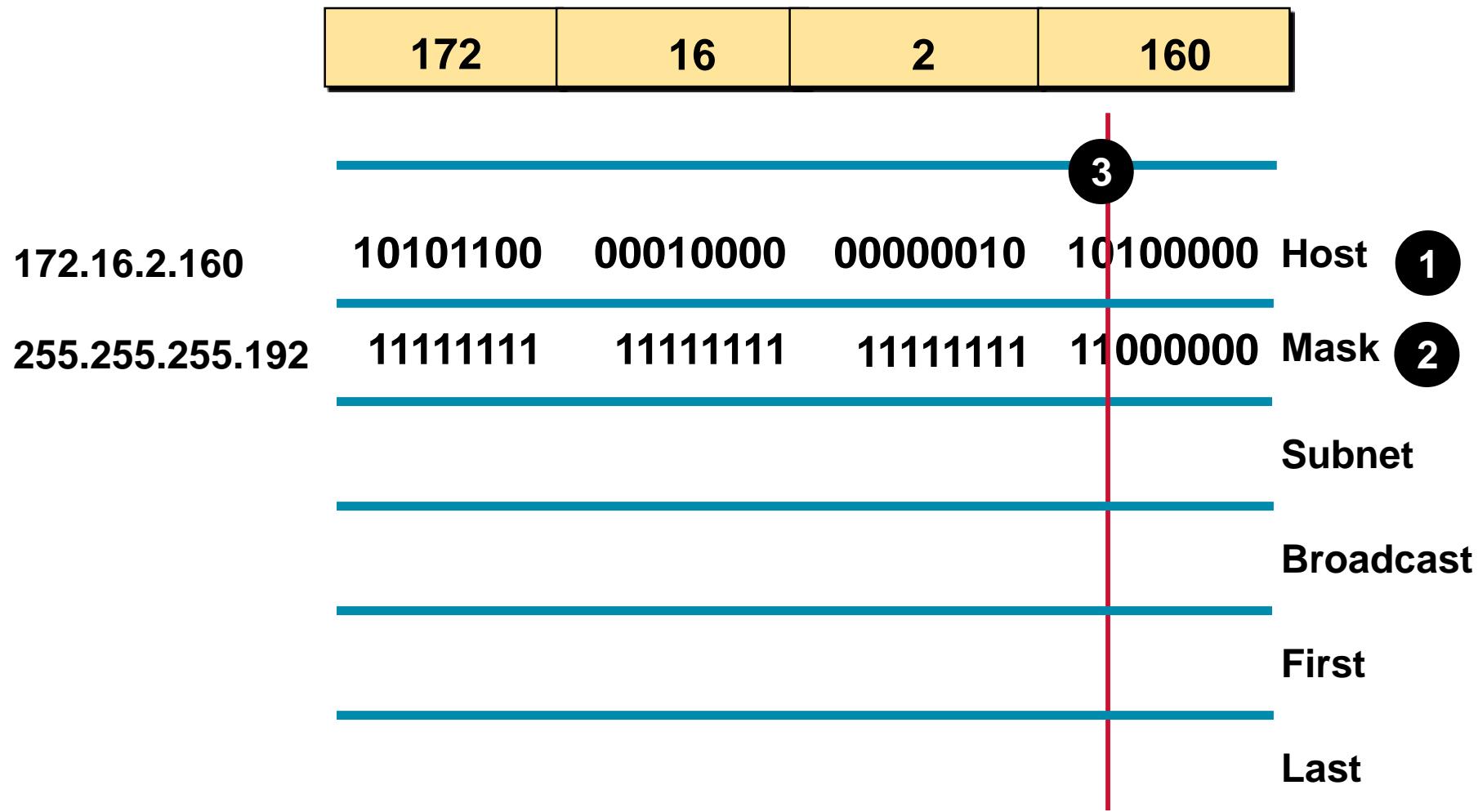


Addressing Summary Example

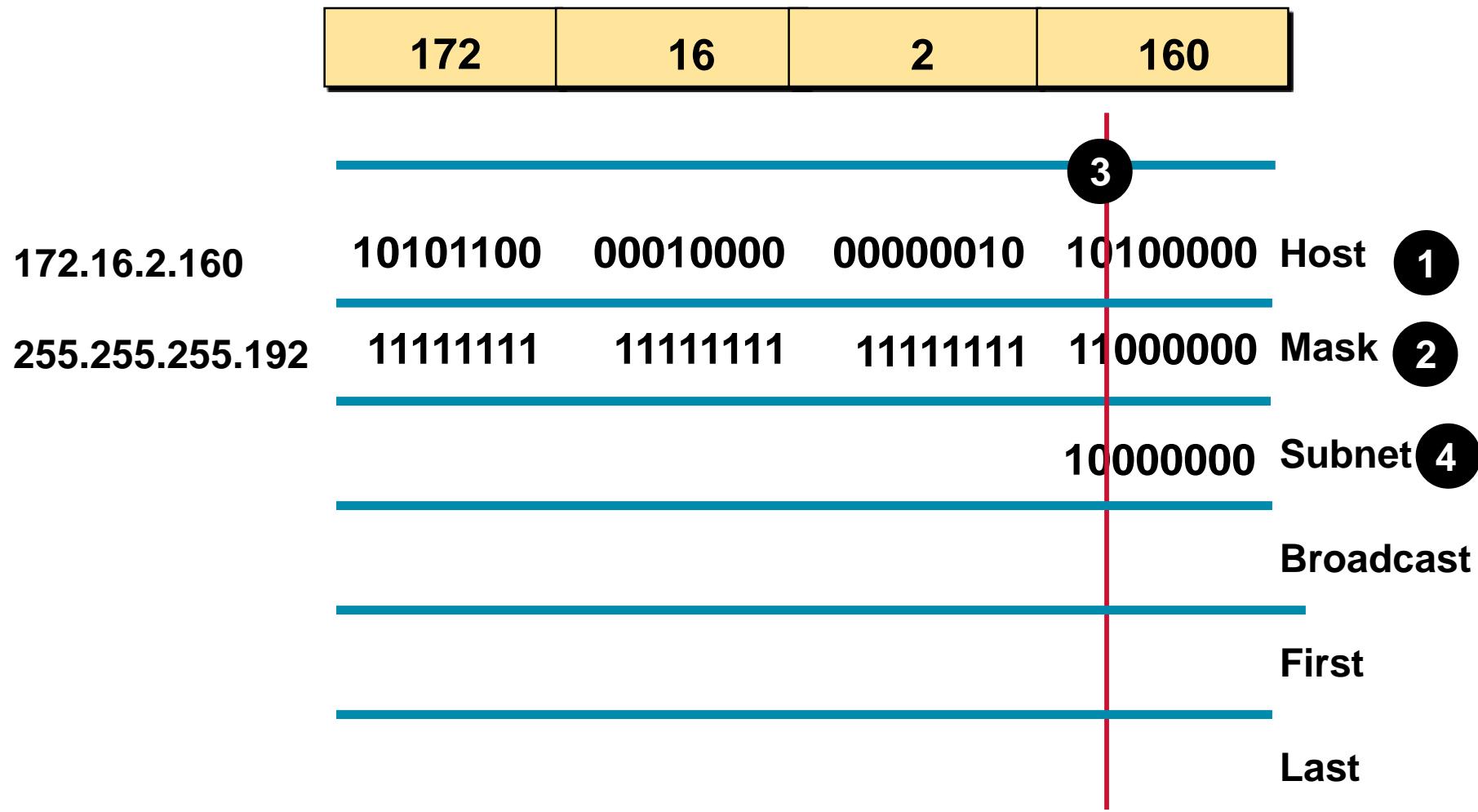
172	16	2	160
-----	----	---	-----



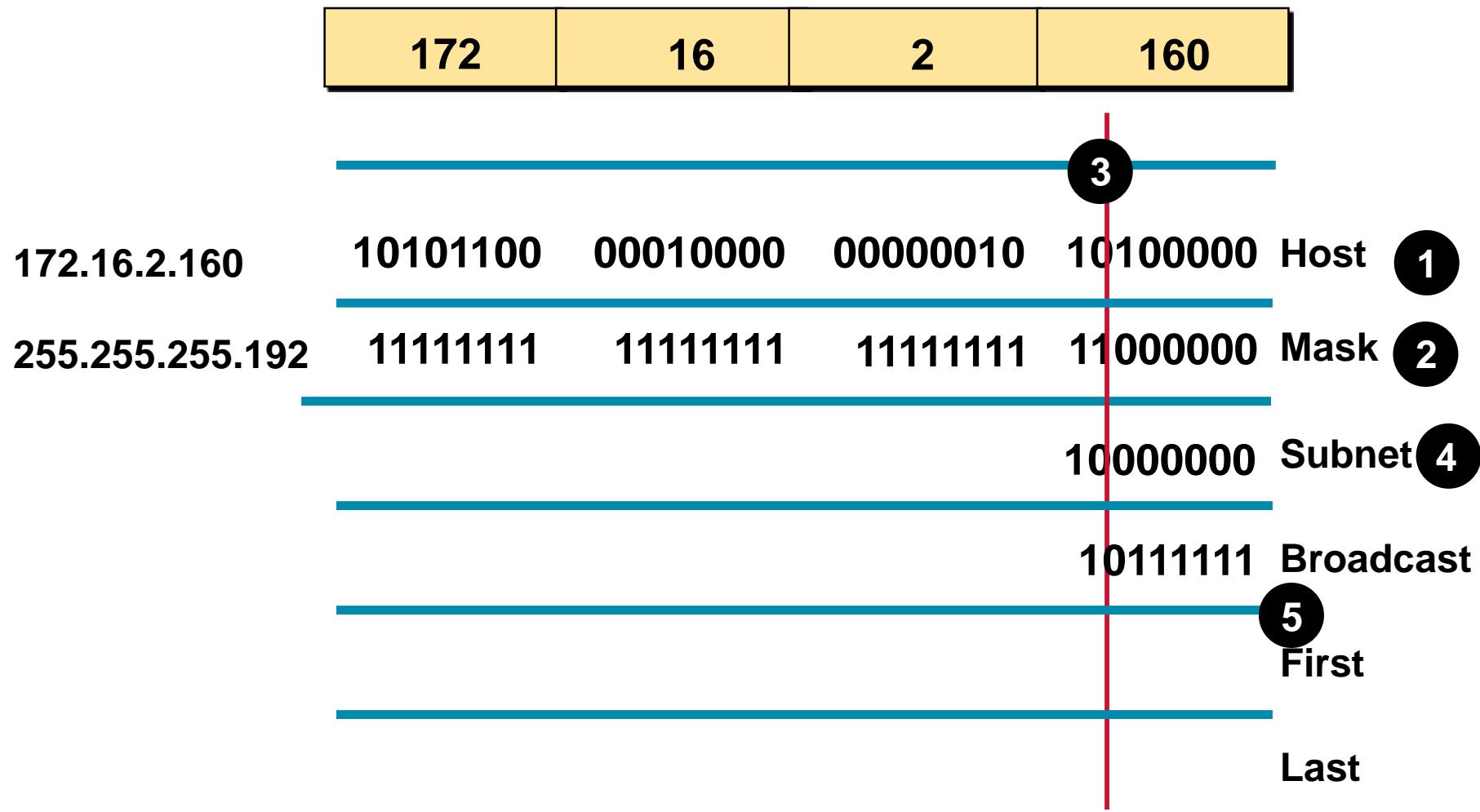
Addressing Summary Example



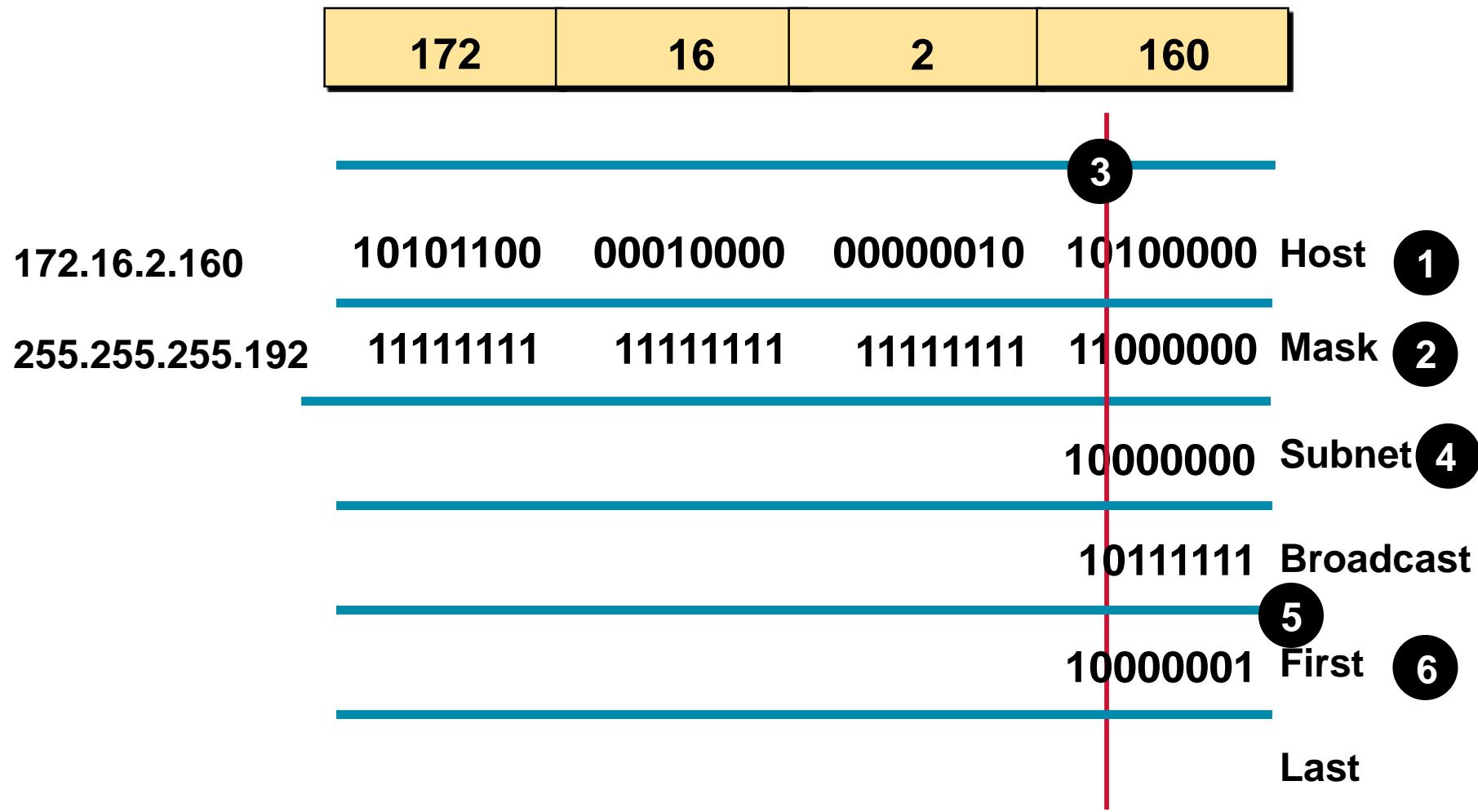
Addressing Summary Example



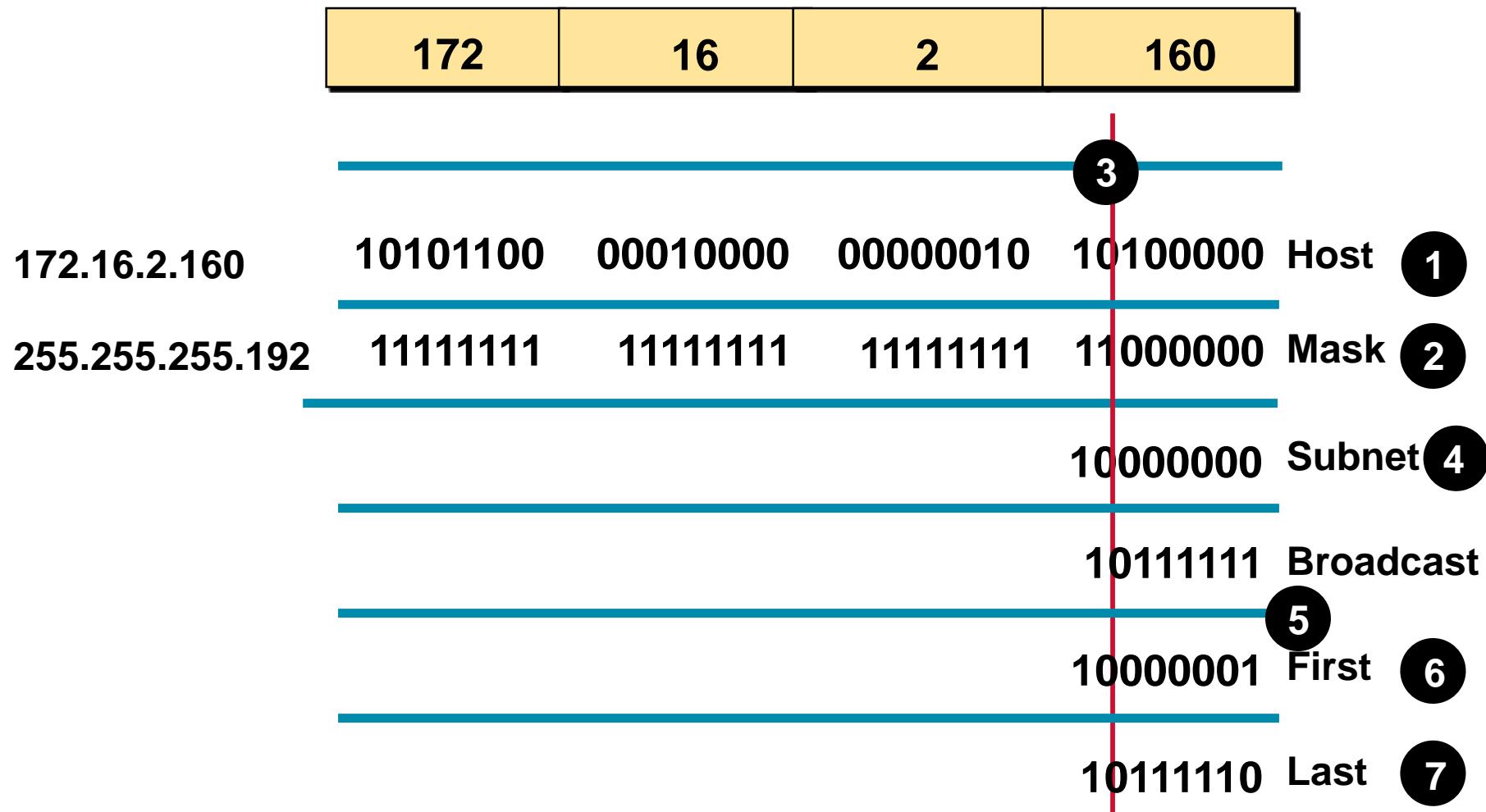
Addressing Summary Example



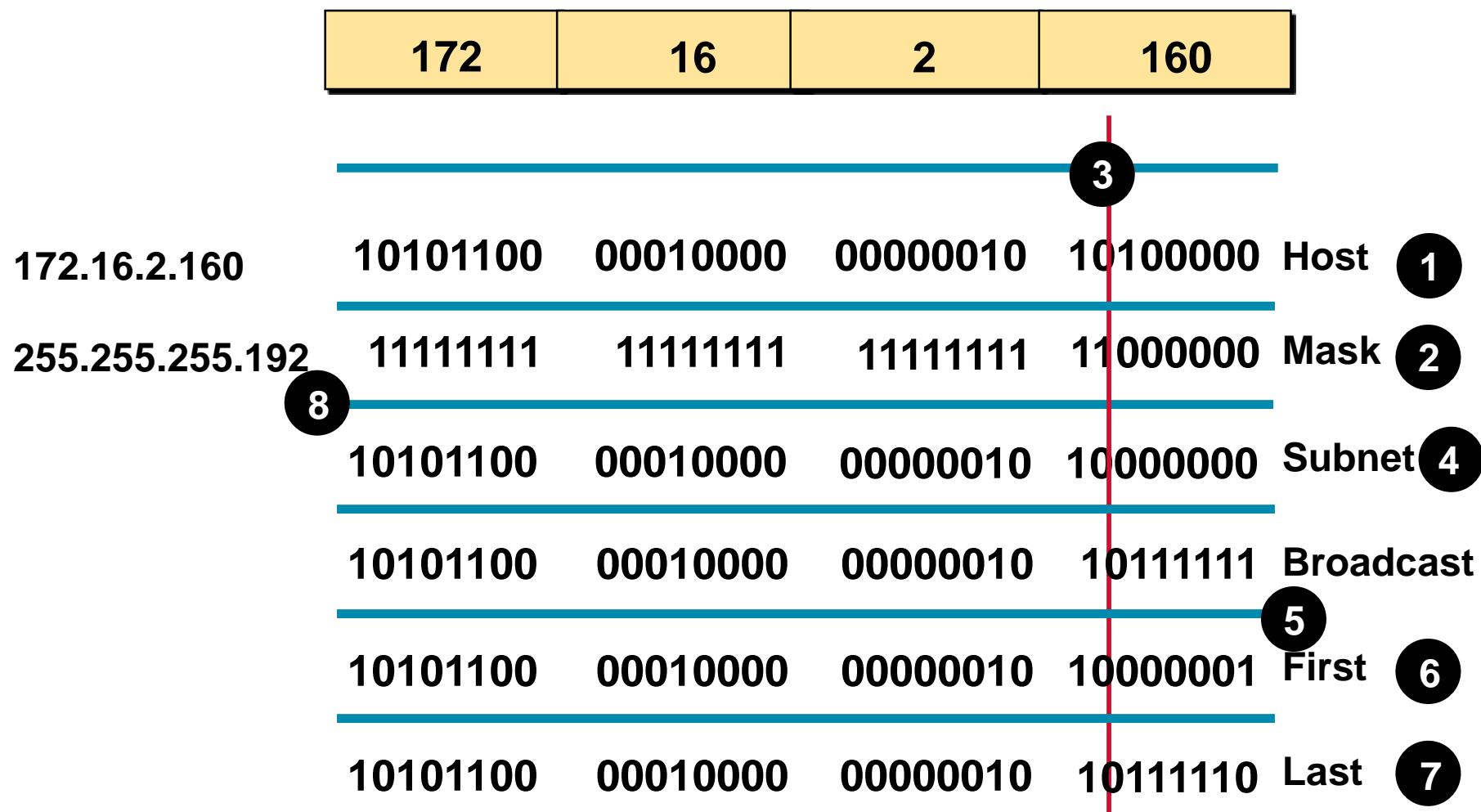
Addressing Summary Example



Addressing Summary Example



Addressing Summary Example



Addressing Summary Example

	172	16	2	160		
172.16.2.160	10101100	00010000	00000010	10100000	Host	1
255.255.255.192	11111111	11111111	11111111	11000000	Mask	2
172.16.2.128	10101100	00010000	00000010	10000000	Subnet	4
172.16.2.191	10101100	00010000	00000010	10111111	Broadcast	
172.16.2.129	10101100	00010000	00000010	10000001	First	6
172.16.2.190	10101100	00010000	00000010	10111110	Last	7

Class B Subnet Example

IP Host Address: 172.16.2.121

Subnet Mask: 255.255.255.0

	Network	Network	Subnet	Host
172.16.2.121:	10101100	00010000	00000010	01111001
255.255.255.0:	11111111	11111111	11111111	00000000
Subnet:	10101100	00010000	00000010	00000000
Broadcast:	10101100	00010000	00000010	11111111

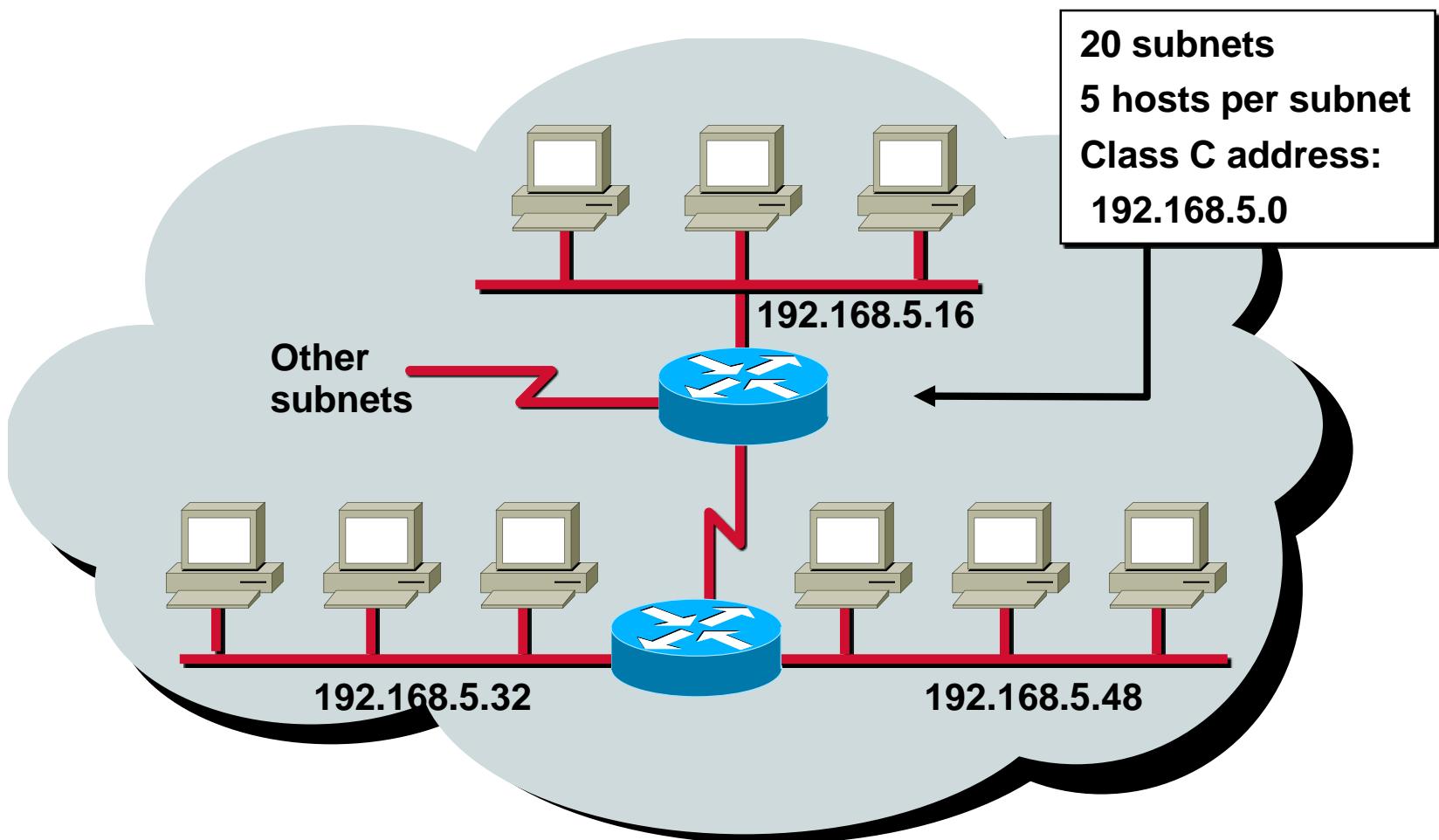
Subnet Address = 172.16.2.0

Host Addresses = 172.16.2.1–172.16.2.254

Broadcast Address = 172.16.2.255

Eight bits of subnetting

Subnet Planning



Class C Subnet Planning Example

IP Host Address: 192.168.5.121

Subnet Mask: 255.255.255.248

Network	Network	Network	Subnet	Host
192.168.5.121:	11000000	10101000	00000101	01111001
255.255.255.248:	11111111	11111111	11111111	11111000
Subnet:	11000000	10101000	00000101	01111000
Broadcast:	11000000	10101000	00000101	01111111

Subnet Address = 192.168.5.120

Host Addresses = 192.168.5.121–192.168.5.126

Broadcast Address = 192.168.5.127

Five Bits of Subnetting

Broadcast Addresses Exercise

Address	Subnet Mask	Class	Subnet	Broadcast
201.222.10.60	255.255.255.248			
15.16.193.6	255.255.248.0			
128.16.32.13	255.255.255.252			
153.50.6.27	255.255.255.128			

Broadcast Addresses Exercise Answers

Address	Subnet Mask	Class	Subnet	Broadcast
201.222.10.60	255.255.255.248	C	201.222.10.56	201.222.10.63
15.16.193.6	255.255.248.0	A	15.16.192.0	15.16.199.255
128.16.32.13	255.255.255.252	B	128.16.32.12	128.16.32.15
153.50.6.27	255.255.255.128	B	153.50.6.0	153.50.6.127

Network Design Fundamentals

Internet Protocol: Subnetting

Presented by,
Jack Crowder, CCIE

Agenda

- Review
 - IP Addressing
 - Format
 - Classfull
 - Reserved
- Subnetting
 - Terms
 - Binary calculations
 - IP subnet calculation

Review

- RFC 1466 (IPv4)
- Address format:
 - 4 octets, dotted decimal notation
 - Example: 192.168.005.100
 - Applied to any interface that wants to communicate in an IP network
- Purpose:
 - Addresses logically grouped: Subnet
 - Routers “deal in” subnets

Classfull Boundaries – 1st Octet

00000000	0	Class A	/
01111111	127		

10000000	128	Class B	/
10111111	191		

11000000	192	Class C	/
11011111	223		

11100000	224	Class D	
11101111	239		

RFC 1878 and 1918

- reserved: 0.0.0.0 – 0.255.255.255 /8
- Class A: 1.0.0.0 - 127.255.255.255
 - reserved: 10.0.0.0 - 10.255.255.255 /8 (private)
 - reserved: 127.0.0.0 - 127.255.255.255 /8 (loopback)
- Class B: 128.0.0.0 - 191.255.255.255
 - reserved: 128.0.0.0 - 128.255.255.255 /8
 - reserved: 172.16.0.0 - 172.31.255.255 /12 (private)
- Class C: 192.0.0.0 - 223.255.255.255
 - reserved: 192.168.0.0 - 192.168.255.255 /16 (private)
- Class D: 224.0.0.0 – 239.255.255.255
 - reserved: 224.0.0.0 - 225.255.255.255 (Multicast)
- reserved: 255.255.255.255 (Broadcast) /32

Subnetting Terms

- Address
- Subnet
- Subnet Mask
- Network field (as determined by the Subnet Mask)
- Host field (as determined by the Subnet Mask)
- Subnet ID: all 0's (in the Host field)
- Broadcast ID: all 1's (in the Host field)
- Unicast (useable address on a subnet)
- Host route

Binary Calculation

2 to the power of X

- 4 Octets (a.k.a. Bytes) in a IPv4 address
 - 8 bits in an octet
 - 32 bits in an address
- 8 Bits in each octet
 - $2^8 = 256$
 - Decimal number range:
0 – 255 (256 numbers counting “0”)
 - Binary range:
00000000 – 11111111 (256 combinations of 1’s and 0’s)

Decimal to Binary

255 = 11111111

128 = 10000000

64 = 01000000

32 = 00100000

16 = 00010000

8 = 00001000

4 = 00000100

2 = 00000010

1 = 00000001

0 = 00000000

255 = 11111111

252 = 11111100

248 = 11111000

240 = 11110000

224 = 11100000

192 = 11000000

128 = 10000000

0 = 00000000

Binary Template

always start counting from 0

$2^x = \underline{\hspace{2cm}} \quad \underline{\hspace{1.5cm}} \quad \underline{\hspace{1.5cm}} \quad \underline{\hspace{1.5cm}} \quad \underline{\hspace{1cm}} \quad \underline{\hspace{1cm}} \quad \underline{\hspace{1cm}} \quad \underline{\hspace{1cm}}$

$128 \quad 64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1$

$x = 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \quad 0$

Position = 8 7 6 5 4 3 2 1

Subnet Mask

- Purpose:
 - Apply the Mask to the IP Address to determine:
 - Network bits
 - Host bits
 - Subnet ID, Broadcast ID & Unicast range
 - Format:
 - 4 octets, dotted decimal notation (same as IP address)
 - Contiguous binary 1's starting from the left
 - Examples:
 - 255.255.255.0 (typical for LAN segment)
 - 255.255.255.252 (typical for WAN pvc)
 - 255.255.255.1 (incorrect)

Subnet Mask in Binary

- 255.255.255.0
- 11111111.11111111.11111111.00000000
- 255.255.255.252
- 11111111.11111111.11111111.11111100
- 255.255.255.1 - incorrect
- 11111111.11111111.11111111.00000001

Subnet Calculation

- Step 1 – Convert:
 - decimal address & mask format to binary address & mask format
- Step 2 – Apply:
 - binary subnet mask to the binary IP address using the “and” function
- Step 3 – Calculate:
 - Subnet ID
 - Broadcast ID
 - Unicast range (usable subnet addresses)

DECIMAL

Step 1

$\text{XXX} = 0 - 255$

XXX.XXX.XXX.XXX



Network bits Host bits

address 185.213.22.219

subnet mask 255.255.255.0

Step 1

XXXXXXX = 0000000 - 1111111

XXXXXXXX.XXXXXXXXXX.XXXXXXXXXX.XXXXXXXXXX

BINARY

Network bits Host bits

10111001.11010101.00010110.11011011

11111111.11111111.11111111.00000000

Step 2: IP Subnet Calculation

Subnet MASK
<AND> IP Address
 IP Subnet

<AND> Rules:

1 and 1 = 1

1 and 0 = 0

0 and 1 = 0

0 and 0 = 0

Another way:

1 and X = X

0 and X = 0

Step 3 – IP Subnet – Example 1

185.213.022.219

255.255.255.000 /24

11111111.11111111.11111111.00000000
_____ .11010101.00010110.11011011

Subnet ID
Broadcast ID

Subnet ID: _____._____._____._____._____

Broadcast ID: _____._____._____._____._____

Unicast: _____._____._____._____._____ - _____

IP Subnet – Example 2

185.213.022.219 255.255.255.252 /30

11111111.11111111.11111111.11111100
_____ .11010101.00010110.11011011

Subnet ID
Broadcast ID

Subnet ID: _____._____._____._____._____

Broadcast ID: _____._____._____._____._____

Unicast: _____._____._____._____._____ - _____

Useable IP Address Calculations

- 1) 32 bits in address
- 2) $32 - \text{network bits} = \text{host bits}$
- 3) $2^{\text{host bits}} = \text{addresses on subnet}$
- 4) addresses - 2 (Broadcast and Subnet ID)
= usable addresses on subnet

255.255.255.240 = /28

$32 - 28 = 4$

$2^4 = 16$ Addresses on Subnet

$16 - 2 = 14$ Unicast addresses on Subnet

Host Route

- Purpose:
 - Used on Loopback and Dial-up interfaces
- Example:
 - 10.5.109.22 255.255.255.255 (/32)

IP Subnet example 3

- IP Address: _____ . _____ . _____ . _____
- Subnet Mask: _____ . _____ . _____ . _____
- Network bits _____ Host bits _____
- Subnet: _____ . _____ . _____ . _____
- Broadcast: _____ . _____ . _____ . _____
- Usable Range: _____ . _____ . _____ . _____
- through _____ . _____ . _____ . _____



Linux Shell Scripting Tutorial Ver. 1.0

Written by Vivek G Gite

I N D E X

- [Introduction](#)
 - [Kernel](#)
 - [Shell](#)
 - [How to use Shell](#)
 - [Common Linux Command Introduction](#)
- [Process](#)
 - [Why Process required](#)
 - [Linux commands related with process](#)
- [Redirection of Standard output/input](#)
 - [Redirectors](#)
 - [Pipes](#)
 - [Filters](#)
- [Shell Programming](#)
 - [Variables in Linux](#)
 - [How to define User defined variables](#)
 - [Rules for Naming variable name](#)
 - [How to print or access value of UDV \(User defined variables\)](#)
 - [How to write shell script](#)
 - [How to Run Shell Scripts](#)
 - [Quotes in Shell Scripts](#)

- [Shell Arithmetic](#)
- [Command Line Processing \(Command Line Arguments\)](#)
- [Why Command Line arguments required](#)
- [Exit Status](#)
- [Filename Shorthand or meta Characters \(i.e. wild cards\)](#)
- [Programming Commands](#)
 - [echo command](#)
 - [Decision making in shell script \(i.e. if command\)](#)
 - [test command or \[expr \]](#)
 - [Loop in shell scripts](#)
 - [The case Statement](#)
 - [The read Statement](#)
- [More Advanced Shell Script Commands](#)
 - [/dev/null - Use to send unwanted output of program](#)
 - [Local and Global Shell variable \(export command\)](#)
 - [Conditional execution i.e. && and ||](#)
 - [I/O Redirection and file descriptors](#)
 - [Functions](#)
 - [User Interface and dialog utility](#)
 - [trap command](#)
 - [getopts command](#)
 - [More examples of Shell Script \(Exercise for You :-\)](#)

© 1998-2000 [FreeOS.com](#) (I) Pvt. Ltd. All rights reserved.

Introduction

This tutorial is designed for beginners only and This tutorial explains the basics of shell programming by showing some examples of shell programs. Its not help or manual for the shell. While reading this tutorial you can find manual quite useful (type man bash at \$ prompt to see manual pages). Manual contains all necessary information you need, but it won't have that much examples, which makes idea more clear. For that reason, this tutorial contains examples rather than all the features of shell. I assumes you have at least working knowledge of Linux i.e. basic commands like how to create, copy, remove files/directories etc or how to use editor like vi or mcedit and login to your system. Before Starting Linux Shell Script Programming you must know

- Kernel
- Shell
- Process
- Redirectors, Pipes, Filters etc.

What's Kernel

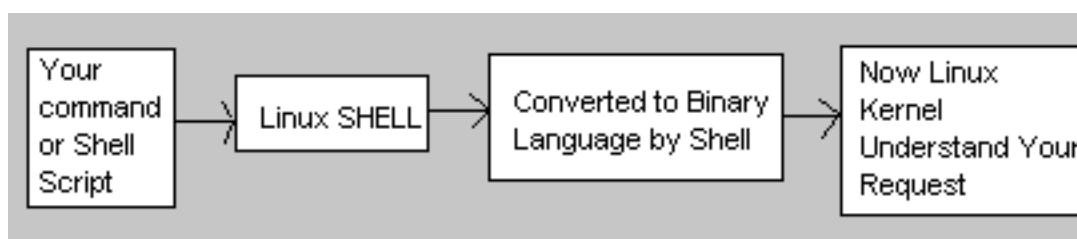
Kernel is hart of Linux O/S. It manages resource of Linux O/S. Resources means facilities available in Linux. For eg. Facility to store data, print data on printer, memory, file management etc . Kernel decides who will use this resource, for how long and when. It runs your programs (or set up to execute binary files) It's Memory resident portion of Linux. It performance following task :-

- I/O management
- Process management
- Device management
- File management
- Memory management

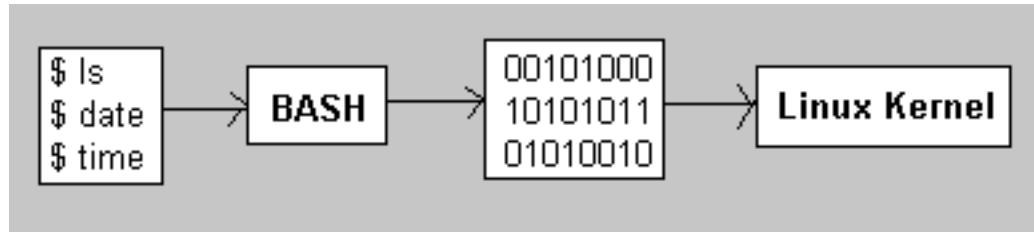
What's Linux Shell

Computer understand the language of 0's and 1's called binary language, In early days of computing, instruction are provided using binary language, which is difficult for all of us, to read and write. So in O/s there is special program called Shell. Shell accepts your instruction or commands in English and translate it into computers native binary language.

This is what Shell Does for US



You type Your command and shell convert it as



It's environment provided for user interaction. Shell is an command language interpreter that executes commands read from the standard input device (keyboard) or from a file. Linux may use one of the following most popular shells (In MS-DOS, Shell name is COMMAND.COM which is also used for same purpose, but it's not as powerful as our Linux Shells are!)

Shell Name	Developed by	Where	Remark
BASH (Bourne-Again SHell)	Brian Fox and Chet Ramey	Free Software Foundation	Most common shell in Linux. It's Freeware shell.
CSH (C SHell)	Bill Joy	University of California (For BSD)	The C shell's syntax and usage are very similar to the C programming language.
KSH (Korn SHell)	David Korn	AT & T Bell Labs	

Any of the above shell reads command from user (via Keyboard or Mouse) and tells Linux O/s what users want. If we are giving commands from keyboard it is called command line interface (Usually in-front of \$ prompt, This prompt is depend upon your shell and Environment that you set or by your System Administrator, therefore you may get different prompt).

NOTE: To find your shell type following command
\$ echo \$SHELL

How to use Shell

To use shell (You start to use your shell as soon as you log into your system) you have to simply type commands. Following is the list of common commands.

Linux Common Commands

NOTE that following commands are for New users or for Beginners only. The purpose is if you use this command you will be more familiar with your shell and secondly, you need some of these command in your Shell script. If you want to get more information or help for this command try following commands For e.g. To see help or options related with date command try

\$ date --help

or To see help or options related with ls command (Here you will screen by screen help, since help of ls command is quite big that can't fit on single screen)

\$ ls --help | more

Syntax: command-name --help

Syntax: man command-name

Syntax: info command-name

See what happened when you type following

\$ man ls

\$ info bash

NOTE: In MS-DOS, you get help by using /? clue or by typing help command as

C:\> dir /?

C:\> date /?

C:\> help time

C:\> help date

C:\> help

Linux Command

For this Purpose	Use this Command Syntax	Example (In front of \$ Prompt)
To see date	date	\$ date
To see who's using system.	who	\$ who
Print working directory	pwd	\$ pwd
List name of files in current directory	ls or dirs	\$ ls
To create text file NOTE: Press and hold CTRL key and press D to stop or to end file (CTRL+D)	cat > { file name }	\$ cat > myfile type your text when done press ^D
To text see files	cat { file name }	\$ cat myfile
To display file one full screen at a time	more { file name }	\$ more myfile
To move or rename file/directory	mv {file1} {file2}	\$ mv sales sales.99
To create multiple file copies with various link. After this both oldfile newfile refers to same name	ln {oldfile} {newfile}	\$ ln Page1 Book1
To remove file	rm file1	\$ rm myfile

Remove all files in given directory/subdirectory. Use it very carefully.	rm -rf {dirname}	\$ rm -rf oldfiles
To change file access permissions u - User who owns the file g - Group file owner o - User classified as other a - All other system user + Set permission - Remove permission r - Read permission w - Write permission x - Execute permission	chmod {u g o a} {+ -} {r w x} {filename}	\$ chmod u+x,g+wx,o+x myscript NOTE: This command set permission for file called 'myscript' as User (Person who creates that file or directory) has execute permission (u+x) Group of file owner can write to this file as well as execute this file (g+wx) Others can only execute file but can not modify it, Since we have not given w (write permission) to them. (o+x).
Read your mail.	mail	\$ mail
To See more about currently login person (i..e. yourself)	who am i	\$ who am i
To login out	logout (OR press CTRL+D)	\$ logout (Note: It may ask you password type your login password, In some case this feature is disabled by System Administrator)
Send mail to other person	mail {user-name}	\$ mail ashish
To count lines, words and characters of given file	wc {file-name}	\$ wc myfile
To searches file for line that match a pattern.	grep {word-to-lookup} {filename}	\$ grep fox myfile
To sort file in following order -r Reverse normal order -n Sort in numeric order -nr Sort in reverse numeric order	sort -r -n -nr {filename}	\$ sort myfile

To print last first line of given file	<code>tail - + {linenumber} {filename}</code>	<code>\$tail +5 myfile</code>
To Use to compare files	<code>cmp {file1} {file2}</code> <code>diff {file1} {file2}</code> OR	<code>\$cmp myfile myfile.old</code>
To print file	<code>pr {file-name}</code>	<code>\$pr myfile</code>

© 1998-2000 FreeOS.com (I) Pvt. Ltd. All rights reserved.

What is Processes

Process is any kind of program or task carried out by your PC. For e.g. \$ ls -lR , is command or a request to list files in a directory and all subdirectory in your current directory. It is a process. A process is program (command given by user) to perform some Job. In Linux when you start process, it gives a number (called PID or process-id), PID starts from 0 to 65535.

Why Process required

Linux is multi-user, multitasking o/s. It means you can run more than two process simultaneously if you wish. For e.g.. To find how many files do you have on your system you may give command like

```
$ ls / -R | wc -l
```

This command will take lot of time to search all files on your system. So you can run such command in Background or simultaneously by giving command like

```
$ ls / -R | wc -l &
```

The ampersand (&) at the end of command tells shells start command (ls / -R | wc -l) and run it in background takes next command immediately. An instance of running command is called process and the number printed by shell is called process-id (PID), this PID can be use to refer specific running process.

Linux Command Related with Process

For this purpose	Use this Command	Example
To see currently running process	ps	\$ ps
To stop any process i.e. to kill process	kill {PID}	\$ kill 1012
To get information about all running process	ps -ag	\$ ps -ag
To stop all process except your shell	kill 0	\$ kill 0
For background processing (With &, use to put particular command and program in background)	linux-command &	\$ ls / -R wc -l &

NOTE that you can only kill process which are created by yourself. A Administrator can almost kill 95-98% process. But some process can not be killed, such as VDU Process.

Redirection of Standard output/input or Input - Output redirection

Mostly all command gives output on screen or take input from keyboard, but in Linux it's possible to send output to file or to read input from file. For e.g. \$ ls command gives output to screen; to send output to file of ls give command , \$ ls > filename. It means put output of ls command to filename. There are three main redirection symbols >, >>, <

(1) > Redirector Symbol

Syntax: Linux-command > filename

To output Linux-commands result to file. Note that If file already exist, it will be overwritten else new file is created. For e.g. To send output of ls command give \$ ls > myfiles

Now if 'myfiles' file exist in your current directory it will be overwritten without any type of warning. (What if I want to send output to file, which is already exist and want to keep information of that file without loosing previous information/data?, For this Read next redirector)

(2) >> Redirector Symbol

Syntax: Linux-command >> filename

To output Linux-commands result to END of file. Note that If file exist , it will be opened and new information / data will be written to END of file, without losing previous information/data, And if file is not exist, then new file is created. For e.g. To send output of date command to already exist file give

\$ date >> myfiles

(3) < Redirector Symbol

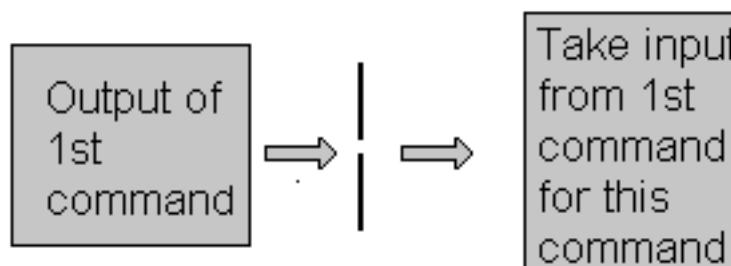
Syntax: Linux-command < filename

To take input to Linux-command from file instead of key-board. For e.g. To take input for cat command give

\$ cat < myfiles

Pips

A pipe is a way to connect the output of one program to the input of another program without any temporary file.



A pipe is nothing but a temporary storage place where the output of one command is stored and then passed as the input for second command. Pipes are used to run more than two commands (Multiple commands) from same command line.

Syntax: command1 | command2

Command using Pips	Meaning or Use of Pipes
\$ ls more	Here the output of ls command is given as input to more command So that output is printed one screen full page at a time
\$ who sort	Here output of who command is given as input to sort command So that it will print sorted list of users
\$ who wc -l	Here output of who command is given as input to wc command So that it will number of user who logon to system
\$ ls -l wc -l	Here output of ls command is given as input to wc command So that it will print number of files in current directory.
\$ who grep raju	Here output of who command is given as input to grep command So that it will print if particular user name if he is logon or nothing is printed (To see for particular user logon)

Filter

If a Linux command accepts its input from the standard input and produces its output on standard output is known as a filter. A filter performs some kind of process on the input and gives output. For e.g.. Suppose we have file called 'hotel.txt' with 100 lines data, And from 'hotel.txt' we would like to print contains from line number 20 to line number 30 and store this result to file called 'hlist' then give command

```
$ tail +20 < hotel.txt | head -n30 > hlist
```

Here head is filter which takes its input from tail command (tail command starts selecting from line number 20 of given file i.e. hotel.txt) and passes this lines to input to head, whose output is redirected to 'hlist' file.

Introduction to Shell Programming

Shell program is series of Linux commands. Shell script is just like batch file is MS-DOS but have more power than the MS-DOS batch file. Shell script can take input from user, file and output them on screen. Useful to create our own commands that can save our lots of time and to automate some task of day today life.

Variables in Linux

Sometimes to process our data/information, it must be kept in computers RAM memory. RAM memory is divided into small locations, and each location had unique number called memory location/address, which is used to hold our data. Programmer can give a unique name to this memory location/address called memory variable or variable (Its a named storage location that may take different values, but only one at a time). In Linux, there are two types of variable

- 1) System variables - Created and maintained by Linux itself. This type of variable defined in CAPITAL LETTERS.
- 2) User defined variables (UDV) - Created and maintained by user. This type of variable defined in lower LETTERS.

Some System variables

You can see system variables by giving command like \$ set, Some of the important System variables are

System Variable	Meaning
BASH=/bin/bash	Our shell name
BASH_VERSION=1.14.7(1)	Our shell version name
COLUMNS=80	No. of columns for our screen
HOME=/home/vivek	Our home directory
LINES=25	No. of rows for our screen
LOGNAME=students	Our logging name
OSTYPE=Linux	Our o/s type : -)
PATH=/usr/bin:/sbin:/bin:/usr/sbin	Our path settings
PS1=[\u@\h \W]\\$	Our prompt settings
PWD=/home/students/Common	Our current working directory
SHELL=/bin/bash	Our shell name
USERNAME=vivek	User name who is currently login to this PC

NOTE that Some of the above settings can be different in your PC. You can print any of the above variables contain as follows

\$ echo \$USERNAME

\$ echo \$HOME

Caution: Do not modify System variable this can some time create problems.

How to define User defined variables (UDV)

To define UDV use following syntax

Syntax: `variablename=value`

NOTE: Here 'value' is assigned to given 'variablename' and Value must be on right side = sign For e.g.

\$ no=10 # this is ok

\$ 10=no # Error, NOT Ok, Value must be on right side of = sign.

To define variable called 'vech' having value Bus

```
$ vech=Bus
To define variable called n having value 10
$ n= 10
```

Rules for Naming variable name (Both UDV and System Variable)

(1) Variable name must begin with Alphanumeric character or underscore character (_), followed by one or more Alphanumeric character. For e.g. Valid shell variable are as follows

```
HOME
SYSTEM_VERSION
vech
no
```

(2) Don't put spaces on either side of the equal sign when assigning value to variable. For e.g.. In following variable declaration there will be no error

```
$ no= 10
But here there will be problem for following
$ no = 10
$ no= 10
$ no = 10
```

(3) Variables are case-sensitive, just like filename in Linux. For e.g.

```
$ no= 10
$ No= 11
$ NO= 20
$ nO= 2
```

Above all are different variable name, so to print value 20 we have to use \$ echo \$NO and Not any of the following

```
$ echo $no      # will print 10 but not 20
$ echo $No      # will print 11 but not 20
$ echo $nO      # will print 2 but not 20
```

(4) You can define NULL variable as follows (NULL variable is variable which has no value at the time of definition) For e.g.

```
$ vech=
$ vech= ""
```

Try to print it's value \$ echo \$vech , Here nothing will be shown because variable has no value i.e. NULL variable.

(5) Do not use ?, * etc, to name your variable names.

How to print or access value of UDV (User defined variables)

To print or access UDV use following syntax

Syntax: \$variableName

For eg. To print contains of variable 'vech'

```
$ echo $vech
```

It will print 'Bus' (if previously defined as vech=Bus) ,To print contains of variable 'n' \$ echo \$n

It will print '10' (if previously defined as n= 10)

Caution: Do not try \$ echo vech It will print vech instead its value 'Bus' and \$ echo n, It will print n instead its value '10', You must use \$ followed by variable name.

Q.1. How to Define variable x with value 10 and print it on screen

```
$ x= 10
$ echo $x
```

Q.2.How to Define variable xn with value Rani and print it on screen

```
$ xn=Rani
$ echo $xn
```

Q.3.How to print sum of two numbers, let's say 6 and 3

```
$ echo 6 + 3
```

This will print $6 + 3$, not the sum 9, To do sum or math operations in shell use expr, syntax is as follows Syntax: `expr op1 operator op2`

Where, op1 and op2 are any Integer Number (Number without decimal point) and operator can be

- + Addition

- Subtraction

- / Division

% Modular, to find remainder For e.g. $20 / 3 = 6$, to find remainder $20 \% 3 = 2$, (Remember its integer calculation)

- * Multiplication

```
$ expr 6 * 3
```

Now It will print sum as 9 , But

```
$ expr 6+3
```

will not work because space is required between number and operator (See Shell Arithmetic)

Q.4.How to define two variable x=20, y=5 and then to print division of x and y (i.e. x/y)

```
$x=20
$ y=5
$ expr x / y
```

Q.5.Modify above and store division of x and y to variable called z

```
$ x=20
$ y=5
$ z=`expr x / y`
$ echo $z
```

Note : For third statement, read Shell Arithmetic.

How to write shell script

Now we write our first script that will print "Knowledge is Power" on screen. To write shell script you can use in of the Linux's text editor such as vi or mcedit or even you can use cat command. Here we are using cat command you can use any of the above text editor. First type following cat command and rest of text as its

```
$ cat > first
#
# My first shell script
#
clear
echo "Knowledge is Power"
```

Press Ctrl + D to save. Now our script is ready. To execute it type command

```
$ ./first
```

This will give error since we have not set Execute permission for our script first; to do this type command

```
$ chmod +x first
$ ./first
```

First screen will be clear, then Knowledge is Power is printed on screen. To print message of variables contains we user echo command, general form of echo command is as follows

```
echo "Message"
echo "Message variable1, variable2....variableN"
```

How to Run Shell Scripts

Because of security of files, in Linux, the creator of Shell Script does not get execution permission by default. So if we wish to run shell script we have to do two things as follows

(1) Use chmod command as follows to give execution permission to our script

Syntax: chmod +x shell-script-name

OR Syntax: chmod 777 shell-script-name

(2) Run our script as

Syntax: ./your-shell-program-name

For e.g.

\$./first

Here '.'(dot) is command, and used in conjunction with shell script. The dot(.) indicates to current shell that the command following the dot(.) has to be executed in the same shell i.e. without the loading of another shell in memory. Or you can also try following syntax to run Shell Script

Syntax: bash &nbsh;&nbsh; your-shell-program-name

OR /bin/sh &nbsh;&nbsh; your-shell-program-name

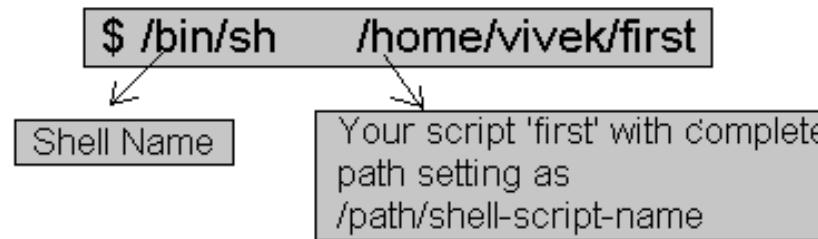
For e.g.

\$ bash first

\$ /bin/sh first

Note that to run script, you need to have in same directory where you created your script, if you are in different directory your script will not run (because of path settings), For eg. Your home directory is (use \$ pwd to see current working directory) /home/vivek. Then you created one script called 'first', after creation of this script you moved to some other directory lets say /home/vivek/Letters/Personal, Now if you try to execute your script it will not run, since script 'first' is in /home/vivek directory, to Overcome this problem there are two ways First, specify complete path of your script when ever you want to run it from other directories like giving following command

\$ /bin/sh /home/vivek/first



Now every time you have to give all this detailed as you work in other directory, this take time and you have to remember complete path. There is another way, if you notice that all of our programs (in form of executable files) are marked as executable and can be directly executed from prompt from any directory (To see executables of our normal program give command \$ ls -l /bin or ls -l /usr/bin) by typing command like

\$ bc

\$ cc myprg.c

\$ cal

etc, How this happed? All our executables files are installed in directory called /bin and /bin directory is set in your PATH setting, Now when you type name of any command at \$ prompt, what shell do is it first look that command in its internal part (called as internal command, which is part of Shell itself, and always available to execute, since they do not need extra executable file), if found as internal command shell will execute it, If not found It will look for current directory, if found shell will execute command from current directory, if not found, then Shell will Look PATH setting, and try to find our requested commands executable file in all of the directories mentioned in PATH settings, if found it will execute it, otherwise it will give message "bash: xxxx :command not found", Still there is one question remain can I run my shell script same as these executables. Yes you can, for

this purpose create bin directory in your home directory and then copy your tested version of shell script to this bin directory. After this you can run your script as executable file without using \$./shell script-name syntax, Following are steps

```
$ cd
$ mkdir bin
$ cp first ~ /bin
$ first
```

Each of above command Explanation

Each of above command	Explanation
\$ cd	Go to your home directory
\$ mkdir bin	Now created bin directory, to install your own shell script, so that script can be run as independent program or can be accessed from any directory
\$ cp first ~ /bin	copy your script 'first' to your bin directory
\$ first	Test whether script is running or not (It will run)

In shell script comment is given with # character. This comments are ignored by your shell. Comments are used to indicate use of script or person who creates/maintained script, or for some programming explanation etc. Remember always set Execute permission for your script.

Commands Related with Shell Programming

(1) echo [options] [string, variables...]

Displays text or variables value on screen.

Options

-n Do not output the trailing new line.

-e Enable interpretation of the following backslash escaped characters in the strings:

\a alert (bell)

\b backspace

\c suppress trailing new line

\n new line

\r carriage return

\t horizontal tab

\\\ backslash

For eg. \$ echo -e "An apple a day keeps away \a\t\tdoctor\n"

(2) More about Quotes

There are three types of quotes

" i.e. Double Quotes

' i.e. Single quotes

` i.e. Back quote

1. "Double Quotes" - Anything enclosed in double quotes removed meaning of those characters (except \ and \$).

2. 'Single quotes' - Enclosed in single quotes remains unchanged.

3. `Back quote` - To execute command.

For eg.

\$ echo "Today is date"

Can't print message with today's date.

\$ echo "Today is `date`".

Now it will print today's date as, Today is Tue Jan, See the `date` statement uses back quote, (See also Shell Arithmetic NOTE).

(3) Shell Arithmetic

Use to perform arithmetic operations For e.g.

```
$ expr 1 + 3
$ expr 2 - 1
$ expr 10 / 2
$ expr 20 % 3 # remainder read as 20 mod 3 and remainder is 2)
$ expr 10 \* 3 # Multiplication use \* not * since its wild card)
$ echo `expr 6 + 3`
```

For the last statement note the following points

- 1) First, before expr keyword we used ` (back quote) sign not the (single quote i.e. ') sign. Back quote is generally found on the key under tilde (~) on PC keyboards OR To the above of TAB key.
- 2) Second, expr is also end with ` i.e. back quote.
- 3) Here expr 6 + 3 is evaluated to 9, then echo command prints 9 as sum
- 4) Here if you use double quote or single quote, it will NOT work, For eg.

```
$ echo "expr 6 + 3" # It will print expr 6 + 3
$ echo 'expr 6 + 3'
```

Command Line Processing

Now try following command (assumes that the file "grate_stories_of" is not exist on your disk)

```
$ ls grate_stories_of
```

It will print message something like -

```
grate_stories_of: No such file or directory
```

Well as it turns out ls was the name of an actual command and shell executed this command when given the command. Now it creates one question What are commands? What happened when you type \$ ls grate_stories_of? The first word on command line, ls, is name of the command to be executed. Everything else on command line is taken as arguments to this command. For eg.

```
$ tail +10 myf
```

Here the name of command is tail, and the arguments are +10 and myf.

Now try to determine command and arguments from following commands:

```
$ ls    foo
$ cp   y   y.bak
$ mv   y.bak   y.okay
$ tail -10   myf
$ mail  raj
$ sort -r -n   myf
$ date
$ clear
```

Command	No. of argument to this command	Actual Argument
ls	1	foo
cp	2	y and y.bak
mv	2	y.bak and y.okay
tail	2	-10 and myf
mail	1	raj
sort	3	-r, -n, and myf
date	0	
clear	0	

NOTE: \$# holds number of arguments specified on command line. and \$* or \$@ refer to all arguments in passed to script. Now to obtain total no. of Argument to particular script, your \$# variable.

Why Command Line arguments required

Let's take rm command, which is used to remove file, But which file you want to remove and how you will you tail this to rm command (Even rm command does not ask you name of file that would like to remove). So what we do is we write as command as follows

```
$ rm {file-name}
```

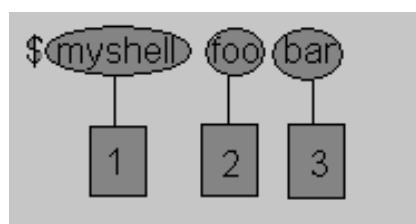
Here rm is command and file-name is file which you would like to remove. This way you tail to rm command which file you would like to remove. So we are doing one way communication with our command by specifying file-name. Also you can pass command line arguments to your script to make it more users friendly. But how we address or access command line argument in our script.

Lets take ls command

```
$ ls -a /*
```

This command has 2 command line argument -a and /* is another. For shell script,

```
$ myshell foo bar
```



1 Shell Script name i.e. myshell

2 First command line argument passed to myshell i.e. foo

3 Second command line argument passed to myshell i.e. bar

In shell if we wish to refer this command line argument we refer above as follows

1 myshell it is \$0

2 foo it is \$1

3 bar it is \$2

Here \$# will be 2 (Since foo and bar only two Arguments), Please note At a time such 9 arguments can be used from \$0..\$9, You can also refer all of them by using \$* (which expand to ` \$0,\$1,\$2...\$9`) Now try to write following for commands, Shell Script Name (\$0), No. of Arguments (i.e. \$#), And actual argument (i.e. \$1,\$2 etc)

```
$ sum 11 20
$ math 4 - 7
$ d
$ bp -5 myf +20
$ ls *
$ cal
$ findBS 4 8 24 BIG
```

Shell Script Name	No. Of Arguments to script	Actual Argument (\$1,..\$9)				
\$0	\$#	\$0	\$1	\$2	\$3	\$4
sum	2	11	20			
math	3	4	-	7		
d	0					

bp	3	-5	myf	+20		
ls	1	*				
cal	0					
findBS	4	4	8	24	BIG	

For e.g. now will write script to print command line argument and we will see how to access them

```
$ cat > demo
#!/bin/sh
#
# Script that demos, command line args
#
echo "Total number of command line argument are $#"
echo "$0 is script name"
echo "$1 is first argument"
echo $2 is second argument"
echo "All of them are :- $* "
```

Save the above script by pressing **ctrl+d**, now make it executable

```
$ chmod +x demo
$ ./demo Hello World
$ cp demo ~ /bin
$ demo
```

Note: After this, For any script you have to used above command, in sequence, I am not going to show you all of the above.

(5) Exit Status

By default in Linux if particular command is executed, it return two type of values, (Values are used to see whether command is successful or not) if return value is zero (0), command is successful, if return value is nonzero (>0), command is not successful or some sort of error executing command/shell script. This value is known as Exit Status of that command. To determine this exit Status we use \$? variable of shell. For eg.

```
$ rm unknow1file
```

It will show error as follows

```
rm: cannot remove `unkowm1file': No such file or directory
and after that if you give command $ echo $?
```

it will print nonzero value(>0) to indicate error. Now give command

```
$ ls
$ echo $?
```

It will print 0 to indicate command is successful. Try the following commands and note down their exit status

```
$ expr 1 + 3
$ echo $?
```

```
$ echo Welcome
$ echo $?
```

```
$ wildwest canwork?
$ echo $?
```

```
$ date
$ echo $?
```

```
$ echon $?
```

\$ echo \$?

(6) **if-then-fi for decision making is shell script** Before making any decision in Shell script you must know following things Type bc at \$ prompt to start Linux calculator program

\$ bc

After this command bc is started and waiting for you commands, i.e. give it some calculation as follows type $5 + 2$ as

$5 + 2$

7

7 is response of bc i.e. addition of $5 + 2$ you can even try

$5 - 2$

$5 / 2$

Now what happened if you type $5 > 2$ as follows

$5 > 2$

0

0 (Zero) is response of bc, How? Here it compare 5 with 2 as, Is 5 is greater than 2, (If I ask same question to you, your answer will be YES) In Linux (bc) gives this 'YES' answer by showing 0 (Zero) value. It means when ever there is any type of comparison in Linux Shell It gives only two answer one is YES and NO is other.

Linux Shell Value	Meaning	Example
Zero Value (0)	Yes/True	0
NON-ZERO Value (> 0)	No/False	-1, 32, 55 anything but not zero

Try following in bc to clear your Idea and not down bc's response

$5 > 12$

$5 == 10$

$5 != 2$

$5 == 5$

$12 < 2$

Expression	Meaning to us	Your Answer	BC's Response (i.e. Linux Shell representation in zero & non-zero value)
$5 > 12$	Is 5 greater than 12	NO	0
$5 == 10$	Is 5 is equal to 10	NO	0
$5 != 2$	Is 5 is NOT equal to 2	YES	1
$5 == 5$	Is 5 is equal to 5	YES	1
$1 < 2$	Is 1 is less than 2	Yes	1

Now will see, if condition which is used for decision making in shell script, If given condition is true then command1 is executed.

Syntax:

```
if condition
then
```

command1 if condition is true or if exit status
of condition is 0 (zero)

...

...

fi

Here condition is nothing but comparison between two values, for compression we can use test or [expr] statements or even exist status can be also used. An expression is nothing but combination of values, relational operator (such as >, <, <> etc) and mathematical operators (such as +, -, / etc). Following are all examples of expression:

```
5 > 2
3 + 6
3 * 65
a < b
c > 5
c > 5 + 30 - 1
```

Type following command (assumes you have file called foo)

```
$ cat foo
$ echo $?
```

The cat command return zero(0) on successful, this can be used in if condition as follows, Write shell script as

```
$ cat > showfile
#!/bin/sh
#
#Script to print file
#
if cat $1
then
    echo -e "\n\nFile $1, found and successfully echoed"
fi
```

Now run it.

```
$ chmod +x showfile
$./showfile foo
```

Here

```
$ ./showfile foo
```

Our shell script name is showfile(\$0) and foo is argument (which is \$1). Now we compare as follows if cat \$1 (i.e. if cat foo)

Now if cat command finds foo file and if its successfully shown on screen, it means our cat command is successful and its exist status is 0 (indicates success) So our if condition is also true and hence statement echo -e "\n\nFile \$1, found and successfully echoed" is proceed by shell. Now if cat command is not successful then it returns non-zero value (indicates some sort of failure) and this statement echo -e "\n\nFile \$1, found and successfully echoed" is skipped by our shell.

Now try to write answer for following

1) Create following script

```
cat > trmif
#
# Script to test rm command and exist status
#
if rm $1
then
    echo "$1 file deleted"
fi
```

(Press Ctrl + d to save)

```
$ chmod +x trmif
```

Now answer the following

A) There is file called foo, on your disk and you give command, \$./trmif foo what will be output.

- B) If bar file not present on your disk and you give command, \$./trmfi bar what will be output.
 C) And if you type \$./trmfi, What will be output.

(7) test command or [expr]

test command or [expr] is used to see if an expression is true, and if it is true it return zero(0), otherwise returns nonzero(>0) for false. Syntax: test expression OR [expression]

Now will write script that determine whether given argument number is positive. Write script as follows

```
$ cat > ispositive
#!/bin/sh
#
# Script to see whether argument is positive
#
if test $1 -gt 0
then
  echo "$1 number is positive"
fi
```

Run it as follows

```
$ chmod +x ispositive
```

```
$ ispositive 5
```

Here o/p : 5 number is positive

```
$ ispositive -45
```

Here o/p : Nothing is printed

```
$ ispositive
```

Here o/p : ./ispositive: test: -gt: unary operator expected

The line, if test \$1 -gt 0 , test to see if first command line argument(\$1) is greater than 0. If it is true(0) then test will return 0 and output will printed as 5 number is positive but for -45 argument there is no output because our condition is not true(0) (no -45 is not greater than 0) hence echo statement is skipped. And for last statement we have not supplied any argument hence error

./ispositive: test: -gt: unary operator expected is generated by shell , to avoid such error we can test whether command line argument is supplied or not. (See command 8 Script example). test or [expr] works with

1. Integer (Number without decimal point)

2. File types

3. Character strings

For Mathematics use following operator in Shell Script

Mathematical Operator in Shell Script	Meaning	Normal Arithmetical/Mathematical Statements	But in Shell	
			For test statement with if command	For [expr] statement with if command
-eq	is equal to	5 == 6	if test 5 -eq 6	if expr [5 -eq 6]
-ne	is not equal to	5 != 6	if test 5 -ne 6	if expr [5 -ne 6]
-lt	is less than	5 < 6	if test 5 -lt 6	if expr [5 -lt 6]
-le	is less than or equal to	5 <= 6	if test 5 -le 6	if expr [5 -le 6]
-gt	is greater than	5 > 6	if test 5 -gt 6	if expr [5 -gt 6]
-ge	is greater than or equal to	5 >= 6	if test 5 -ge 6	if expr [5 -ge 6]

NOTE: == is equal, != is not equal.

For string Comparisons use

Operator	Meaning
string1 = string2	string1 is equal to string2
string1 != string2	string1 is NOT equal to string2
string1	string1 is NOT NULL or not defined
-n string1	string1 is NOT NULL and does exist
-z string1	string1 is NULL and does exist

Shell also test for file and directory types

Test	Meaning
-s file	Non empty file
-f file	Is File exist or normal file and not a directory
-d dir	Is Directory exist and not a file
-w file	Is writeable file
-r file	Is read-only file
-x file	Is file is executable

Logical Operators

Logical operators are used to combine two or more condition at a time

Operator	Meaning
! expression	Logical NOT
expression1 -a expression2	Logical AND
expression1 -o expression2	Logical OR

(8)if...else...fi

If given condition is true then command1 is executed otherwise command2 is executed.

Syntax:

```

if condition
then
    command1 if condition is true or if exit status
    of condition is 0(zero)
    ...
    ...
else
    command2 if condition is false or if exit status
    of condition is >0 (nonzero)
    ...
    ...
fi

```

For eg. Write Script as follows

```
$ cat > isnump_n
#!/bin/sh
#
```

```
# Script to see whether argument is positive or negative
#
if [ $# -eq 0 ]
then
    echo "$0 : You must give/supply one integers"
    exit 1
fi

if test $1 -gt 0
then
    echo "$1 number is positive"
else
    echo "$1 number is negative"
fi
```

Try it as follows

```
$ chmod +x isnump_n
$ isnump_n 5
Here o/p : 5 number is positive
$ isnump_n -45
Here o/p : -45 number is negative
$ isnump_n
Here o/p : ./ispos_n : You must give/supply one integers
$ isnump_n 0
Here o/p : 0 number is negative
```

Here first we see if no command line argument is given then it print error message as "./ispos_n : You must give/supply one integers". if statement checks whether number of argument (\$#) passed to script is not equal (-eq) to 0, if we passed any argument to script then this if statement is false and if no command line argument is given then this if statement is true. The echo command i.e. echo "\$0 : You must give/supply one integers"



1 will print Name of script

2 will print this error message

And finally statement exit 1 causes normal program termination with exit status 1 (nonzero means script is not successfully run), The last sample run \$ isnump_n 0 , gives output as "0 number is negative", because given argument is not > 0, hence condition is false and it's taken as negative number. To avoid this replace second if statement with if test \$1 -ge 0.

(9) Multilevel if-then-else

Syntax:

```
if condition
then
    condition is zero (true - 0)
    execute all commands up to elif statement
elif condition1
    condition1 is zero (true - 0)
    execute all commands up to elif statement
elif condition2
    condition2 is zero (true - 0)
    execute all commands up to elif statement
```

```

else
    None of the above condition, condition1, condition2 are true (i.e.
    all of the above nonzero or false)
    execute all commands up to fi
fi

```

For e.g. Write script as \$ cat > elf #!/bin/sh # # Script to test if..elif...else # # if [\$1 -gt 0] then echo "\$1 is positive" elif [\$1 -lt 0] then echo "\$1 is negative" elif [\$1 -eq 0] then echo "\$1 is zero" else echo "Opps! \$1 is not number, give number" fi Try above script with \$ chmod +x elf \$./elf 1 \$./elf -2 \$./elf 0 \$./elf a Here o/p for last sample run: ./elf: [: -gt: unary operator expected ./elf: [: -lt: unary operator expected ./elf: [: -eq: unary operator expected Opps! a is not number, give number Above program gives error for last run, here integer comparison is expected therefore error like "./elf: [: -gt: unary operator expected" occurs, but still our program notify this thing to user by providing message "Opps! a is not number, give number". (10)Loops in Shell Scripts

Computer can repeat particular instruction again and again, until particular condition satisfies. A group of instruction that is executed repeatedly is called a loop.

(a) for loop Syntax:

```

for { variable name } in { list }
do
    execute one for each item in the list until the list is
    not finished (And repeat all statement between do and done)
done

```

Suppose,

```
$ cat > testfor
for i in 1 2 3 4 5
do
    echo "Welcome $i times"
done
```

Run it as,

```
$ chmod +x testfor
$ ./testfor
```

The for loop first creates i variable and assigned a number to i from the list of number from 1 to 5, The shell execute echo statement for each assignment of i. (This is usually know as iteration) This process will continue until all the items in the list were not finished, because of this it will repeat 5 echo statements. for e.g. Now try script as follows

```
$ cat > mtable
#!/bin/sh
#
#Script to test for loop
#
#
if [ $# -eq 0 ]
then
    echo "Error - Number missing form command line argument"
    echo "Syntax : $0 number"
    echo " Use to print multiplication table for given number"
    exit 1
fi
n= $1
for i in 1 2 3 4 5 6 7 8 9 10
```

```
do
  echo "$n * $i = `expr $i \* $n`"
done
```

Save and Run it as

```
$ chmod +x mtable
$ ./mtable 7
$ ./mtable
```

For first run, Above program print multiplication table of given number where i = 1,2 ... 10 is multiply by given n (here command line argument 7) in order to produce multiplication table as

```
7 * 1 = 7
7 * 2 = 14
```

...

..
7 * 10 = 70 And for Second run, it will print message -

Error - Number missing form command line argument

Syntax : ./mtable number

 Use to print multiplication table for given number

This happened because we have not supplied given number for which we want multiplication table, Hence we are showing Error message, Syntax and usage of our script. This is good idea if our program takes some argument, let the user know what is use of this script and how to used it. Note that to terminate our script we used 'exit 1' command which takes 1 as argument (1Indicates error and therefore script is terminated)

(b)while loop

Syntax:

```
while [ condition ]
do
  command1
  command2
  command3
  ...
  ....
done
```

Loop is executed as long as given condition is true. For eg. Above for loop program can be written using while loop as

```
$cat > nt1
#!/bin/sh
#
#Script to test while statement
#
#
if [ $# -eq 0 ]
then
  echo "Error - Number missing from command line argument"
  echo "Syntax : $0 number"
  echo " Use to print multiplication table for given number"
  exit 1
fi
n=$1
i=1
while [ $i -le 10 ]
```

```

do
  echo "$n * $i = `expr $i \* $n`"
  i=`expr $i + 1`
done

```

Save it and try as

```

$ chmod +x nt1
$./nt1 7

```

Above loop can be explained as follows

n=\$1	Set the value of command line argument to variable n. (Here it's set to 7)
i=1	Set variable i to 1
while [\$i -le 10]	This is our loop condition, here if value of i is less than 10 then, shell execute all statements between do and done
do	Start loop
echo "\$n * \$i = `expr \$i * \$n`"	Print multiplication table as 7 * 1 = 7 7 * 2 = 14 7 * 10 = 70, Here each time value of variable n is multiply be i.
i=`expr \$i + 1`	Increment i by 1 and store result to i. (i.e. i=i+1) Caution: If we ignore (remove) this statement than our loop become infinite loop because value of variable i always remain less than 10 and program will only output 7 * 1 = 7 E (infinite times)
done	Loop stops here if i is not less than 10 i.e. condition of loop is not true. Hence loop is terminated.

From the above discussion not following points about loops

- (a) First, the variable used in loop condition must be initialized, Next execution of the loop begins.
- (b) A test (condition) is made at the beginning of each iteration.
- (c) The body of loop ends with a statement that modifies the value of the test (condition) variable.

(11) The case Statement

The case statement is good alternative to Multilevel if-then-else-fi statement. It enable you to match several values against one variable. Its easier to read and write.

Syntax:

```

case $variable-name in
  pattern1)   command
              ...
              ...
              command;;
  pattern2)   command

```

```

    ...
    ..
    command;;
patternN)   command
    ...
    ..
    command;;
* )          command
    ...
    ..
    command;;
esac

```

The \$variable-name is compared against the patterns until a match is found. The shell then executes all the statements up to the two semicolons that are next to each other. The default is *) and its executed if no match is found. For eg. Create script as follows

```

$ cat > car
#
# if no vehicle name is given
# i.e. -z $1 is defined and it is NULL
#
# if no command line arg

if [ -z $1 ]
then
    rental= "*** Unknown vehicle ***"
elif [ -n $1 ]
then
# otherwise make first arg as rental
    rental=$1
fi

case $rental in
    "car") echo "For $rental Rs.20 per k/m";;
    "van") echo "For $rental Rs.10 per k/m";;
    "jeep") echo "For $rental Rs.5 per k/m";;
    "bicycle") echo "For $rental 20 paisa per k/m";;
    *) echo "Sorry, I can not get a $rental for you";;
esac

```

Save it by pressing CTRL+D

```

$ chmod +x car
$ car van
$ car car
$ car Maruti-800

```

Here first we will check, that if \$1(first command line argument) is not given set value of rental variable to "*** Unknown vehicle ***", if value given then set it to given value. The \$rental is compared against the patterns until a match is found. Here for first run its match with van and it will show output For van Rs.10 per k/m. For second run it prints, "For car Rs.20 per k/m". And for last run, there is no match for Maruti-800, hence default i.e. *) is executed and it prints, "Sorry, I can not get a Maruti-800 for you". Note that esac is always required to indicate end of case statement.

(12) The read Statement

Use to get input from keyboard and store them to variable.

Syntax: read variable1, variable2,...variableN**Create script as**

```
$ cat > sayH
#
#Script to read your name from key-board
#
echo "Your first name please:"
read fname
echo "Hello $fname, Lets be friend!"
```

Run it as follows

```
$ chmod +x sayH
$ ./sayH
```

This script first ask you your name and then waits to enter name from the user, Then user enters name from keyboard (After giving name you have to press ENTER key) and this entered name through keyboard is stored (assigned) to variable fname.

(13)Filename Shorthand or meta Characters (i.e. wild cards)

* or ? or [...] is one of such shorthand character.

* Matches any string or group of characters.

For e.g. \$ ls * , will show all files, \$ ls a* - will show all files whose first name is starting with letter 'a', \$ ls *.c ,will show all files having extension .c \$ ls ut*.c, will show all files having extension .c but first two letters of file name must be 'ut'.

? Matches any single character.

For e.g. \$ ls ?, will show one single letter file name, \$ ls fo?, will show all files whose names are 3 character long and file name begin with fo

[...] Matches any one of the enclosed characters.

For e.g. \$ ls [abc]* - will show all files beginning with letters a,b,c

[...-] A pair of characters separated by a minus sign denotes a range;

For eg. \$ ls /bin/[a-c]* - will show all files name beginning with letter a,b or c like

/bin/arch	/bin/awk	/bin/bsh	/bin/chmod	/bin/cp
/bin/ash	/bin basename	/bin/cat	/bin/chown	/bin/cpio
/bin/ash.static	/bin/bash	/bin/chgrp	/bin/consolechars	/bin/csh

But

```
$ ls /bin/[!a-o]
```

```
$ ls /bin/[ ^ a-o]
```

If the first character following the [is a ! or a ^ then any character not enclosed is matched i.e. do not show us file name that beginning with a,b,c,e...o, like

/bin/ps	/bin/rvi	/bin/sleep	/bin/touch	/bin/view
/bin/pwd	/bin/rview	/bin/sort	/bin/true	/bin/wcomp
/bin/red	/bin/sayHello	/bin/stty	/bin/umount	/bin/xconf
/bin/remadmin	/bin/sed	/bin/su	/bin/uname	/bin/ypdomainname
/bin/rm	/bin/setserial	/bin/sync	/bin/userconf	/bin/zcat
/bin/rmdir	/bin/sfxload	/bin/tar	/bin/usleep	
/bin/rpm	/bin/sh	/bin/tcsh	/bin/vi	

(14)command1;command2

To run two command with one command line. For eg. \$ date;who , Will print today's date followed

by users who are currently login. Note that You can't use \$ date who for same purpose, you must put semicolon in between date and who command.

© 1998-2000 [FreeOS.com](#) (I) Pvt. Ltd. All rights reserved.

More Advanced Shell Script Commands

/dev/null - Use to send unwanted output of program

This is special Linux file which is used to send any unwanted output from program/command.

Syntax: command > /dev/null

For e.g. \$ ls > /dev/null , output of this command is not shown on screen its send to this special file. The /dev directory contains other device files. The files in this directory mostly represent peripheral devices such disks like floppy disk, sound card, line printers etc.

Local and Global Shell variable (export command)

Normally all our variables are local. Local variable can be used in same shell, if you load another copy of shell (by typing the /bin/bash at the \$ prompt) then new shell ignores all old shell's variable. For e.g.

Consider following example

```
$ vech=Bus
$ echo $vech
Bus
$ /bin/bash
$ echo $vech
```

NOTE:-Empty line printed

```
$ vech=Car
$ echo $vech
Car
$ exit
$ echo $vech
Bus
```

Command	Meaning
\$ vech=Bus	Create new local variable 'vech' with Bus as value in first shell
\$ echo \$vech	Print the contents of variable vech
\$ /bin/bash	Now load second shell in memory (Which ignores all old shell's variable)
\$ echo \$vech	Print the contents of variable vech
\$ vech=Car	Create new local variable 'vech' with Car as value in second shell
\$ echo \$vech	Print the contents of variable vech
\$ exit	Exit from second shell return to first shell
\$ echo \$vech	Print the contents of variable vech (Now you can see first shells variable and its value)

We can copy old shell's variable to new shell (i.e. first shell's variable to second shell), such variable is known as Global Shell variable. To do this use export command

Syntax: export variable1, variable2,.....variableN

For e.g.

```
$ vech=Bus
$ echo $vech
Bus
$ export vech
$ /bin/bash
$ echo $vech
Bus
$ exit
$ echo $vech
```

Bus

Command	Meaning
\$ vech=Bus	Create new local variable 'vech' with Bus as value in first shell
\$ echo \$vech	Print the contains of variable vech
\$ export vech	Export first shells variable to second shell
\$ /bin/bash	Now load second shell in memory (Old shell's variable is accessed from second shell, <i>if they are exported</i>)
\$ echo \$vech	Print the contains of variable vech
\$ exit	Exit from second shell return to first shell
\$ echo \$vech	Print the contains of variable vech

Conditional execution i.e. && and ||

The control operators are && (read as AND) and || (read as OR). An AND list has the

Syntax: command1 && command2

Here command2 is executed if, and only if, command1 returns an exit status of zero. An OR list has the

Syntax: command1 || command2

Here command2 is executed if and only if command1 returns a non-zero exit status. You can use both as follows

command1 && comamnd2 if exist status is zero || command3 if exit status is non-zero

Here if command1 is executed successfully then shell will run command2 and if command1 is not successful then command3 is executed. For e.g.

\$ rm myf && echo File is removed successfully || echo File is not removed

If file (myf) is removed successful (exist status is zero) then "echo File is removed successfully" statement is executed, otherwise "echo File is not removed" statement is executed (since exist status is non-zero)

I/O Redirection and file descriptors

As you know I/O redirectors are used to send output of command to file or to read input from file. (See Input/Output redirection). Now consider following examples

\$ cat > myf

This is my file
^D

Above command send output of cat command to myf file

\$ cal

Above command prints calendar on screen, but if you wish to store this calendar to file then give command

\$ cal > mycal

The cal command send output to mycal file. This is called output redirection

\$ sort

10

-20

11

2

^D

-20

2

10

11

Here sort command takes input from keyboard and then sorts the number, If we wish to take input from file give command as follows

\$ cat > nos

10

```
-20
11
2
^D
$ sort < nos
-20
2
10
11
```

First we have created the file nos, then we have taken input from this file and sort command prints sorted numbers. This is called input redirection. In Linux (And in C programming Language) your keyboard, screen etc are treated as files. Following are name of such files

Standard File	File Descriptors number	Use	Example
stdin	0	as Standard input	Keyboard
stdout	1	as Standard output	Screen
stderr	2	as Standard error	Screen

By default in Linux every program has three files associated with it, (when we start our program these three files are automatically opened by your shell) The use of first two files (i.e. stdin and stdout) , are already seen by us. The last file stderr (numbered as 2) is used by our program to print error on screen. You can redirect the output from a file descriptor directly to file with following

Syntax: file-descriptor-number>filename

For e.g.

```
$ rm bad_file_name111
```

rm: cannot remove `bad_file_name111': No such file or directory ,is the output (error) of the above program. Now if we try to redirect this error-output to file, it can not be send to file

```
$ rm bad_file_name111 > er
```

Still it prints output on stderr as rm: cannot remove `bad_file_name111': No such file or directory, And if you see er file as \$ cat er , This file is empty, since output is send to error device and you can not redirect it to copy this error-output to your file 'er'. To overcome this we have to use following command

```
$ rm bad_file_name111 2>er
```

Note that no space are allowed between 2 and >, The 2>er directs the standard error output to file. 2 number is default number of stderr file. Now consider another example, here we are writing shell script as follows

```
$ cat > demoscr
if [ $# -ne 2 ]
then
echo "Error : Number are not supplied"
echo "Usage : $0 number1 number2"
exit 1
fi
ans=`expr $1 + $2`
echo "Sum is $ans"
```

Try it as follows

```
$ chmod +x demoscr
```

```
$ ./demoscr
```

```
Error : Number are not supplied
Usage : ./demoscr number1 number2
$ ./demoscr > er1
$ ./demoscr 5 7
Sum is 12
```

Here for first sample run , our script prints error message indicating that we have not given two number. For second sample run, we have redirect output of our script to file, since it's error we have to show it to

user, It means we have to print our error message on stderr not on stdout. To overcome this problem replace above echo statements as follows

```
echo "Error : Number are not supplied" 1>&2
echo "Usage : $0 number1 number2" 1>&2
```

Now if you run as

```
$ ./demoscr > er1
```

Error : Number are not supplied

Usage : ./demoscr number1 number2

It will print error message on stderr and not on stdout. The `1>&2` at the end of echo statement, directs the standard output (stdout) to standard error (stderr) device.

Syntax: from>&destination

Functions

Function is series of instruction/commands. Function performs particular activity in shell. To define function use following

Syntax:

```
function-name ( )
{
    command1
    command2
    .....
    ...
    commandN
    return
}
```

Where function-name is name of your function, that executes these commands. A return statement will terminate the function. For e.g. Type `SayHello()` at \$ prompt as follows

```
$ SayHello()
{
echo "Hello $LOGNAME, Have nice computing"
return
}
```

Now to execute this `SayHello()` function just type its name as follows

```
$ SayHello
```

Hello xxxxx, Have nice computing

This way you can call your function. Note that after restarting your computer you will lose this `SayHello()` function, since it's created for that session only. To overcome this problem and to add your own function to automat some of the day today life task, your function to `/etc/bashrc` file. Note that to add function to this file you must logon as root. Following is the sample `/etc/bashrc` file with `today()` function , which is used to print formatted date. First logon as root or if you already logon with your name (your login is not root), and want to move to root account, then you can type following command , when asked for password type root (administrators) password

```
$ su
```

password:

Now open file as (Note your prompt is changed to # from \$ to indicate you are root)

```
# vi /etc/bashrc
```

OR

```
# mcedit /etc/bashrc
```

At the end of file add following in `/etc/bashrc` file

```
#
# today() to print formatted date
#
# To run this function type today at the $ prompt
# Added by Vivek to show function in Linux
#
```

```
today()
{
echo This is a ` date + "%A %d in %B of %Y (%r)" `
return
}
```

Save the file and exit it, after all this modification your file may look like as follows

```
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile

# For some unknown reason bash refuses to inherit
# PS1 in some circumstances that I can't figure out.
# Putting PS1 here ensures that it gets loaded every time.

PS1="[\u@\h \W]\$\n"

#
# today() to print formatted date
#
# To run this function type today at the $ prompt
# Added by Vivek to show function in Linux
today()
{
echo This is a ` date + "%A %d in %B of %Y (%r)" `
return
}
```

To run function first completely logout by typing exit at the \$prompt (Or press CTRL + D, Note you may have to type exit (CTRL + D) twice if you login to root account by using su command) ,then login and type \$ today , this way today() is available to all user in your system, If you want to add particular function to particular user then open .bashrc file in your home directory as follows

```
# vi .bashrc
OR
# mcedit .bashrc
At the end of file add following in .bashrc file
SayBuy()
{
    echo "Buy $LOGNAME ! Life never be the same, until you log again!"
    echo "Press a key to logout. . ."
    read
    return
}
```

Save the file and exit it, after all this modification your file may look like as follows

```
# .bashrc
#
# User specific aliases and functions
# Source global definitions

if [ -f /etc/bashrc ]; then
. /etc/bashrc
fi

SayBuy()
```

```
{
echo "Buy $LOGNAME ! Life never be the same, until you log again!"
echo "Press a key to logout. . ."
read
return
}
```

To run function first logout by typing exit at the \$ prompt (Or press CTRL + D) ,then logon and type \$ SayBuy , this way SayBuy() is available to only in your login and not to all user in system, Use .bashrc file in your home directory to add User specific aliases and functions only. (Tip: If you want to show some message or want to perform some action when you logout, Open file .bash_logout in your home directory and add your stuff here For e.g. When ever I logout, I want to show message Buy! Then open your .bash_logout file using text editor such as vi and add statement

```
echo "Buy $LOGNAME, Press a key. . ."
```

```
read
```

Save and exit from the file. Then to test this logout from your system by pressing CTRL + D (or type exit) immediately you will see message "Buy xxxx, Press a key. . .", after pressing key you will be exited.)

User Interface and dialog utility

Good program/shell script must interact with users. There are two ways to this one is use command line to script when you want input, second use statement like echo and read to read input into variable from the prompt. For e.g. Write script as

```
$ cat > userinte
#
# Script to demo echo and read command for user interaction
#
echo "Your good name please :"
read na
echo "Your age please :"
read age
neyr=`expr $age + 1`
echo "Hello $na, next year you will be $neyr yrs old."
```

Save it and run as

```
$ chmod +x userinte
$ ./userinte
```

Your good name please :

Vivek

Your age please :

25

Hello Vivek, next year you will be 26 yrs old.

Even you can create menus to interact with user, first show menu option, then ask user to choose menu item, and take appropriate action according to selected menu item, this technique is show in following script

```
$ cat > menuui
#
# Script to create simple menus and take action according to that selected
# menu item
#
while :
do
    clear
    echo "-----"
    echo " Main Menu "
    echo "-----"
    echo "[ 1 ] Show Todays date/time"
    echo "[ 2 ] Show files in current directory"
```

```

echo "[3] Show calendar"
echo "[4] Start editor to write letters"
echo "[5] Exit/Stop"
echo "===== "
echo -n "Enter your menu choice [1-5]: "
read yourch
case $yourch in
    1) echo "Today is `date` , press a key..."; read ;;
    2) echo "Files in `pwd` "; ls -l ; echo "Press a key..."; read ;;
    3) cal ; echo "Press a key..."; read ;;
    4) vi ;;
    5) exit 0 ;;
*) echo "Opps!!! Please select choice 1,2,3,4, or 5";
   echo "Press a key..."; read ;;
esac
done

```

Above all statement explained in following table

Statement	Explanation
while :	Start infinite loop, this loop will only break if you select 5 (i.e. Exit/Stop menu item) as your menu choice
do	Start loop
clear	Clear the screen, each and every time
echo "-----" echo " Main Menu " echo "-----" echo "[1] Show Todays date/time" echo "[2] Show files in current directory" echo "[3] Show calendar" echo "[4] Start editor to write letters" echo "[5] Exit/Stop" echo "===== "	Show menu on screen with menu items
echo -n "Enter your menu choice [1-5]: "	Ask user to enter menu item number
read yourch	Read menu item number from user
case \$yourch in 1) echo "Today is `date` , press a key..."; read ;; 2) echo "Files in `pwd` "; ls -l ; echo "Press a key..."; read ;; 3) cal ; echo "Press a key..."; read ;; 4) vi ;; 5) exit 0 ;; *) echo "Opps!!! Please select choice 1,2,3,4, or 5"; echo "Press a key..."; read ;; esac	Take appropriate action according to selected menu item, If menu item is not between 1 - 5, then show error and ask user to input number between 1-5 again
done	Stop loop , if menu item number is 5 (i.e. Exit/Stop)

User interface usually includes, menus, different type of boxes like info box, message box, Input box etc. In Linux shell there is no built-in facility available to create such user interface, But there is one utility supplied with Red Hat Linux version 6.0 called dialog, which is used to create different type of boxes like info box, message box, menu box, Input box etc. Now try dialog utility as follows :

```
$ cat > dia1
dialog --title "Linux Dialog Utility Infobox" --backtitle "Linux Shell Script \
Tutorial" --infobox "This is dialog box called infobox, which is used \
to show some information on screen, Thanks to Savio Lam and \
Stuart Herbert to give us this utility. Press any key. . ." 7 50 ; read
```

Save the shell script and run as

```
$ chmod +x dia1
$ ./dia1
```

After executing this dialog statement you will see box on screen with titled as "Welcome to Linux Dialog Utility" and message "This is dialog....Press any key. . ." inside this box. The title of box is specified by --title option and info box with --infobox "Message" with this option. Here 7 and 50 are height-of-box and width-of-box respectively. "Linux Shell Script Tutorial" is the backtitle of dialog show on upper left side of screen and below that line is drawn. Use dialog utility to Display dialog boxes from shell scripts.

Syntax:

```
dialog --title {title} --backtitle {backtitle} {Box options}
where Box options can be any one of following
--yesno      {text}  {height} {width}
--msgbox     {text}  {height} {width}
--infobox    {text}  {height} {width}
--inputbox   {text}  {height} {width} [{init}]
--textbox    {file}  {height} {width}
--menu       {text}  {height} {width} {menu} {height} {tag1} item1...
```

msgbox using dialog utility

```
$cat > dia2
dialog --title "Linux Dialog Utility Msgbox" --backtitle "Linux Shell Script \
Tutorial" --msgbox "This is dialog box called msgbox, which is used \
to show some information on screen which has also Ok button, Thanks to Savio Lam \
and Stuart Herbert to give us this utility. Press any key. . ." 9 50
```

Save it and run as

```
$ chmod +x dia2
$ ./dia2
```

yesno box using dialog utility

```
$ cat > dia3
dialog --title "Alert : Delete File" --backtitle "Linux Shell Script \
Tutorial" --yesno "\nDo you want to delete '/usr/letters/jobapplication' \
file" 7 60
sel=$?
case $sel in
  0) echo "You select to delete file";;
  1) echo "You select not to delete file";;
  255) echo "Canceled by you by pressing [ESC] key";;
esac
```

Save it and run as

```
$ chmod +x dia3
$ ./dia3
```

Above script creates yesno type dialog box, which is used to ask some questions to the user , and answer to those question either yes or no. After asking question how do we know, whether user has press yes or no button ? The answer is exit status, if user press yes button exit status will be zero, if user press no button exit status will be one and if user press Escape key to cancel dialog box exit status will be one 255. That is what we have tested in our above shell as

Statement	Meaning
sel=\$?	Get exit status of dialog utility
case \$sel in 0) echo "You select to delete file";; 1) echo "You select not to delete file";; 255) echo "Canceled by you by pressing [Escape] key";; esac	Now take action according to exit status of dialog utility, if exit status is 0 , delete file, if exit status is 1 do not delete file and if exit status is 255, means Escape key is pressed.

inputbox using dialog utility

```
$ cat > dia4
dialog --title "Inputbox - To take input from you" --backtitle "Linux Shell\
Script Tutorial" --inputbox "Enter your name please" 8 60 2>/tmp/input.$$
sel=$?
na=`cat /tmp/input.$$`
case $sel in
    0) echo "Hello $na" ;;
    1) echo "Cancel is Press" ;;
    255) echo "[ESCAPE] key pressed" ;;
esac
rm -f /tmp/input.$$
```

Inputbox is used to take input from user, Here we are taking Name of user as input. But where we are going to store inputted name, the answer is to redirect inputted name to file via statement `2>/tmp/input.$$` at the end of dialog command, which means send screen output to file called `/tmp/input.$$,` letter we can retrieve this inputted name and store to variable as follows
`na= `cat /tmp/input.$$``. For inputbox exit status is as follows

Exit Status for Inputbox	Meaning
0	Command is successful
1	Cancel button is pressed by user
255	Escape key is pressed by user

Now we will write script to create menus using dialog utility, following are menu items

Date/time

Calendar

Editor

and action for each menu-item is follows

MENU-ITEM	ACTION
Date/time	Show current date/time
Calendar	Show calendar
Editor	Start vi Editor

Create script as follows

```
$ cat > smenu
#
#How to create small menu using dialog
#
dialog --backtitle "Linux Shell Script Tutorial" --title "Main\
Menu" --menu "Move using [UP] [DOWN],[Enter] to\
```

```
Select" 15 50 3 \
Date/time    "Shows Date and Time" \
Calendar    "To see calendar " \
Editor       "To start vi editor " 2>/tmp/menuitem.$$

menuitem= `cat /tmp/menuitem.$$` 

opt=$?

case $menuitem in
  Date/time) date;;
  Calendar) cal;;
  Editor) vi;;
esac

rm -f /tmp/menuitem.$$
```

Save it and run as

```
$ chmod +x smenu
$ ./smenu
```

Here --menu option is used of dialog utility to create menus, menu option take

--menu options	Meaning
"Move using [UP] [DOWN],[Enter] to Select"	This is text show before menu
15	Height of box
50	Width of box
3	Height of menu
Date/time "Shows Date and Time"	First menu item called as <i>tag1</i> (i.e. Date/time) and description for menu item called as <i>item1</i> (i.e. "Shows Date and Time")
Calendar "To see calendar "	First menu item called as <i>tag2</i> (i.e. Calendar) and description for menu item called as <i>item2</i> (i.e. "To see calendar")
Editor "To start vi editor "	First menu item called as <i>tag3</i> (i.e. Editor) and description for menu item called as <i>item3</i> (i.e."To start vi editor")
2>/tmp/menuitem.\$\$	Send selected menu item (tag) to this temporary file

After creating menus, user selects menu-item by pressing enter key the selected choice is redirected to temporary file, Next this menu-item is retrieved from temporary file and following case statement compare the menu-item and takes appropriate step according to selected menu item. As you see, dialog utility allows more powerful user interaction then the older read and echo statement. The only problem with dialog utility is it work slowly.

trap command

Now consider following script

```
$ cat > testsign
ls -R /
```

Save and run it as

```
$ chmod +x testsign
$ ./testsign
```

Now if you press **ctrl + c**, while running this script, script get terminated. The **ctrl + c** here work as signal, When such signal occurs its send to all process currently running in your system. Now consider following shell script

```
$ cat > testsign1
#
# Why to trap signal, version 1
#
Take_input1()
{
recno=0
clear
echo "Appointment Note keeper Application for Linux"
echo -n "Enter your database file name : "
read filename

if [ ! -f $filename ]; then
    echo "Sorry, $filename does not exist, Creating $filename database"
    echo "Appointment Note keeper Application database file" > $filename
fi

echo "Data entry start data: `date`" >/tmp/input0.$$
#
# Set a infinite loop
#
while :
do
    echo -n "Appointment Title:"
    read na
    echo -n "Appoint time :"
    read ti
    echo -n "Any Remark :"
    read remark
    echo -n "Is data okay (y/n) ?"
    read ans

if [ $ans = y -o $ans = Y ]; then
    recno=`expr $recno + 1`
    echo "$recno. $na $ti $remark" >> /tmp/input0.$$
fi

    echo -n "Add next appointment (y/n)?"
    read isnext

if [ $isnext = n -o $isnext = N ]; then
    cat /tmp/input0.$$ >> $filename
    rm -f /tmp/input0.$$
    return # terminate loop
fi
done
}

#
#
# Call our user define function : Take_input1
#
Take_input1
```

Save it and run as

```
$ chmod +x testsign1
$ ./testsign1
```

It first ask you main database file where all appointment of that day is stored, if no such database file found, file is created, after that it open one temporary file in /tmp directory, and puts today's date in that

file. Then one infinite loop begins, which ask appointment title, time and remark, if this information is correct its written to temporary file, After that script ask user , whether he/she wants add next appointment record, if yes then next record is added , otherwise all records are copied from temporary file to database file and then loop will be terminated. You can view your database file by using cat command. Now problem is that while running this script, if you press CTRL + C, your shell script gets terminated and temporary file are left in /tmp directory. For e.g. try as follows

```
$./testsign1
```

After given database file name and after adding at least one appointment record to temporary file press CTRL+C, Our script get terminated, and it left temporary file in /tmp directory, you can check this by giving command as follows

```
$ ls /tmp/input*
```

Our script needs to detect when such signal (event) occurs, To achieve this we have to first detect Signal using trap command

Syntax: trap { commands} { signal number list}

Signal Number	When occurs
0	shell exit
1	hangup
2	interrupt (CTRL+C)
3	quit
9	kill (cannot be caught)

To catch this signal in above script, put trap statement before calling Take_input1 function as trap del_file 2 ., Here trap command called del_file() when 2 number interrupt (i.e.CTRL+C) occurs. Open above script in editor and modify it so that at the end it will look like as follows

```
$ vi testsign1
or
$ mcedit testsign1
#
# signal is trapped to delete temporary file , version 2
#
del_file()
{
    echo " * * * CTRL + C Trap Occurs (removing temporary file) * * * "
    rm -f /tmp/input0.$$
    exit 1
}

Take_input1()
{
recno=0
clear
echo "Appointment Note keeper Application for Linux"
echo -n "Enter your database file name : "
read filename

if [ ! -f $filename ]; then
    echo "Sorry, $filename does not exist, Creating $filename database"
    echo "Appointment Note keeper Application database file" > $filename
fi

echo "Data entry start data: `date` " >/tmp/input0.$$
#
# Set a infinite loop
#
while :
do
```

```

echo -n "Appointment Title:"
read na
echo -n "Appoint time :"
read ti
echo -n "Any Remark :"
read remark
echo -n "Is data okay (y/n) ?"
read ans

if [ $ans = y -o $ans = Y ]; then
    recno= `expr $recno + 1`
    echo "$recno. $na $ti $remark" >> /tmp/input0.$$
fi

echo -n "Add next appointment (y/n)?"
read isnext

if [ $isnext = n -o $isnext = N ]; then
    cat /tmp/input0.$$ >> $filename
    rm -f /tmp/input0.$$
    return # terminate loop
fi
done
}

#
# Set trap to for CTRL+C interrupt,
# When occurs it first it calls del_file() and then exit
#
trap del_file 2

#
# Call our user define function : Take_input1
#
Take_input1

```

Now save it run the program as

```
$ ./testsign1
```

After given database file name and after giving appointment title press CTRL+C. Here we have already captured this CTRL + C signal (interrupt), so first our function `del_file()` is called, in which it gives message as " * * * CTRL + C Trap Occurs (removing temporary file)* * * " and then it remove our temporary file and then exit with exit status 1. Now check /tmp directory as follows

```
$ ls /tmp/input*
```

Now Shell will report no such temporary file exit.

getopts command

This command is used to check valid command line argument passed to script. Usually used in while loop.

Syntax: `getopts {optsring} {variable1}`

getopts is used by shell to parse command line argument. optstring contains the option letters to be recognized; if a letter is followed by a colon, the option is expected to have an argument, which should be separated from it by white space. Each time it is invoked, getopts places the next option in the shell variable variable1, When an option requires an argument, getopts places that argument into the variable OPTARG. On errors getopts diagnostic messages are printed when illegal options or missing option arguments are encountered. If an illegal option is seen, getopts places ? into variable1. For e.g. We have script called ani which has syntax as

```
ani -n -a -s -w -d
```

Options: These are optional argument

```
-n name of animal
```

- a age of animal
- s sex of animal
- w weight of animal
- d demo values (if any of the above options are used
their values are not taken)

```
$ cat > ani
#
# Usage: ani -n -a -s -w -d
#
#
# help_ani() To print help
#
help_ani()
{
    echo "Usage: $0 -n -a -s -w -d"
    echo "Options: These are optional argument"
    echo "    -n name of animal"
    echo "    -a age of animal"
    echo "    -s sex of animal"
    echo "    -w weight of animal"
    echo "    -d demo values (if any of the above options are used"
    echo "        their values are not taken)"
    exit 1
}
#
#Start main procedure
#
#
#Set default value for variable
#
isdef=0

na=Moti
age="2 Months"
sex=Male
weight=3Kg

#
#if no argument
#
if [ $# -lt 1 ]; then
    help_ani
fi

while getopts n:a:s:w:d opt
do
    case "$opt" in
        n) na="$OPTARG";;
        a) age="$OPTARG";;
        s) sex="$OPTARG";;
        w) weight="$OPTARG";;
        d) isdef=1;;
        \?) help_ani;;
    esac
done
```

```

if [ $isdef -eq 0 ]
then
    echo "Animal Name: $na, Age: $age, Sex: $sex, Weight: $weight (user define mode)"
else
    na="Pluto Dog"
    age=3
    sex=Male
    weight=20kg
    echo "Animal Name: $na, Age: $age, Sex: $sex, Weight: $weight (demo mode)"
fi

```

Save it and run as follows

```

$ chmod +x ani
$ ani -n Lassie -a 4 -s Female -w 20Kg
$ ani -a 4 -s Female -n Lassie -w 20Kg
$ ani -n Lassie -s Female -w 20Kg -a 4
$ ani -w 20Kg -s Female -n Lassie -a 4
$ ani -w 20Kg -s Female
$ ani -n Lassie -a 4
$ ani -n Lassie
$ ani -a 2

```

See because of getopt, we can pass command line argument in different style. Following are invalid options for ani script

\$ ani -nLassie -a4 -sFemal -w20Kg

Here no space between option and their value.

\$ ani -nLassie-a4-sFemal-w20Kg

\$ ani -n Lassie -a 4 -s Female -w 20Kg -c Mammal

Here -c is not one of the options.

More examples of Shell Script (Exercise for You :-)

First try to write this shell script, as exercise, if any problem or for sample answer to this Shell script open the shell script file supplied with this tutorial.

Q.1. How to write shell script that will add two nos, which are supplied as command line argument, and if these two nos are not given show error and its usage

Answer: See Q1 shell Script.

Q.2. Write Script to find out biggest number from given three nos. Nos are supplies as command line argument. Print error if sufficient arguments are not supplied.

Answer: See Q2 shell Script.

Q.3. Write script to print nos as 5,4,3,2,1 using while loop.

Answer: See Q3 shell Script.

Q.4. Write Script, using case statement to perform basic math operation as follows

+ addition

- subtraction

x multiplication

/ division

The name of script must be 'q4' which works as follows

\$./q4 20 / 3, Also check for sufficient command line arguments

Answer: See Q4 shell Script.

Q.5. Write Script to see current date, time, username, and current directory

Answer: See Q5 shell Script.

Q.6. Write script to print given number in reverse order, for eg. If no is 123 it must print as 321.

Answer: See Q6 shell Script.

Q.7. Write script to print given numbers sum of all digit, For eg. If no is 123 it's sum of all digit will be $1+2+3 = 6$.

Answer: See Q7 shell Script.

Q.8. How to perform real number (number with decimal point) calculation in Linux

Answer: Use Linux's bc command

Q.9. How to calculate $5.12 + 2.5$ real number calculation at \$ prompt in Shell ?

Answer: Use command as , \$ echo $5.12 + 2.5$ | bc , here we are giving echo commands output to bc to calculate the $5.12 + 2.5$

Q.10. How to perform real number calculation in shell script and store result to third variable , lets say a=5.66, b=8.67, c=a+b?

Answer: See Q10 shell Script.

Q.11. Write script to determine whether given file exist or not, file name is supplied as command line argument, also check for sufficient number of command line argument

Answer: See Q11 shell Script.

Q.12. Write script to determine whether given command line argument (\$1) contains "*" symbol or not, if \$1 does not contains "*" symbol add it to \$1, otherwise show message "Symbol is not required". For e.g. If we called this script Q12 then after giving ,

\$ Q12 /bin

Here \$1 is /bin, it should check whether "*" symbol is present or not if not it should print Required i.e. /bin/*, and if symbol present then Symbol is not required must be printed. Test your script as

\$ Q12 /bin

\$ Q12 /bin/*

Answer: See Q12 shell Script

Q.13. Write script to print contains of file from given line number to next given number of lines. For e.g. If we called this script as Q13 and run as

\$ Q13 5 5 myf , Here print contains of 'myf' file from line number 5 to next 5 line of that file.

Answer: See Q13 shell Script

Q.14. Write script to implement getopt statement, your script should understand following command line argument called this script Q14,

Q14 -c -d -m -e

Where options work as

-c clear the screen

-d show list of files in current working directory

-m start mc (midnight commander shell) , if installed

-e { editor } start this { editor } if installed

Answer: See Q14 shell Script

Q.15. Write script called sayHello, put this script into your startup file called .bash_profile, the script should run as soon as you logon to system, and it print any one of the following message in infobox using dialog utility, if installed in your system, If dialog utility is not installed then use echo statement to print message : -

Good Morning

Good Afternoon

Good Evening , according to system time.

Answer: See Q15 shell Script

Q.16. How to write script, that will print, Message "Hello World" , in Bold and Blink effect, and in different colors like red, brown etc using echo command.

Answer: See Q16 shell Script

Q.17. Write script to implement background process that will continually print current time in upper right

corner of the screen , while user can do his/her normal job at \$ prompt.

Answer: See Q17 shell Script.

Q.18. Write shell script to implement menus using dialog utility. Menu-items and action according to select menu-item is as follows

Menu-Item	Purpose	Action for Menu-Item
Date/time	To see current date time	Date and time must be shown using infobox of dialog utility
Calendar	To see current calendar	Calendar must be shown using infobox of dialog utility
Delete	To delete selected file	First ask user name of directory where all files are present, if no name of directory given assumes current directory, then show all files only of that directory, Files must be shown on screen using menus of dialog utility, let the user select the file, then ask the confirmation to user whether he/she wants to delete selected file, if answer is yes then delete the file , report errors if any while deleting file to user.
Exit	To Exit this shell script	Exit/Stops the menu driven program i.e. this script

Note: Create function for all action for e.g. To show date/time on screen create function `show_datetime()`.

Answer: See Q18 shell Script.

Q.19. Write shell script to show various system configuration like

- 1) Currently logged user and his logname
- 2) Your current shell
- 3) Your home directory
- 4) Your operating system type
- 5) Your current path setting
- 6) Your current working directory
- 7) Show Currently logged number of users
- 8) About your os and version ,release number , kernel version
- 9) Show all available shells
- 10) Show mouse settings
- 11) Show computer cpu information like processor type, speed etc
- 12) Show memory information
- 13) Show hard disk information like size of hard-disk, cache memory, model etc
- 14) File system (Mounted)

Answer: See Q19 shell Script.

That's all!, Thanks for reading, This tutorial ends here, but Linux programming environment is big, rich and productive (As you see from above shell script exercise), you should continue to read more advance topics and books. If you have any suggestion or new ideas or problem with this tutorial, please feel free to contact me.

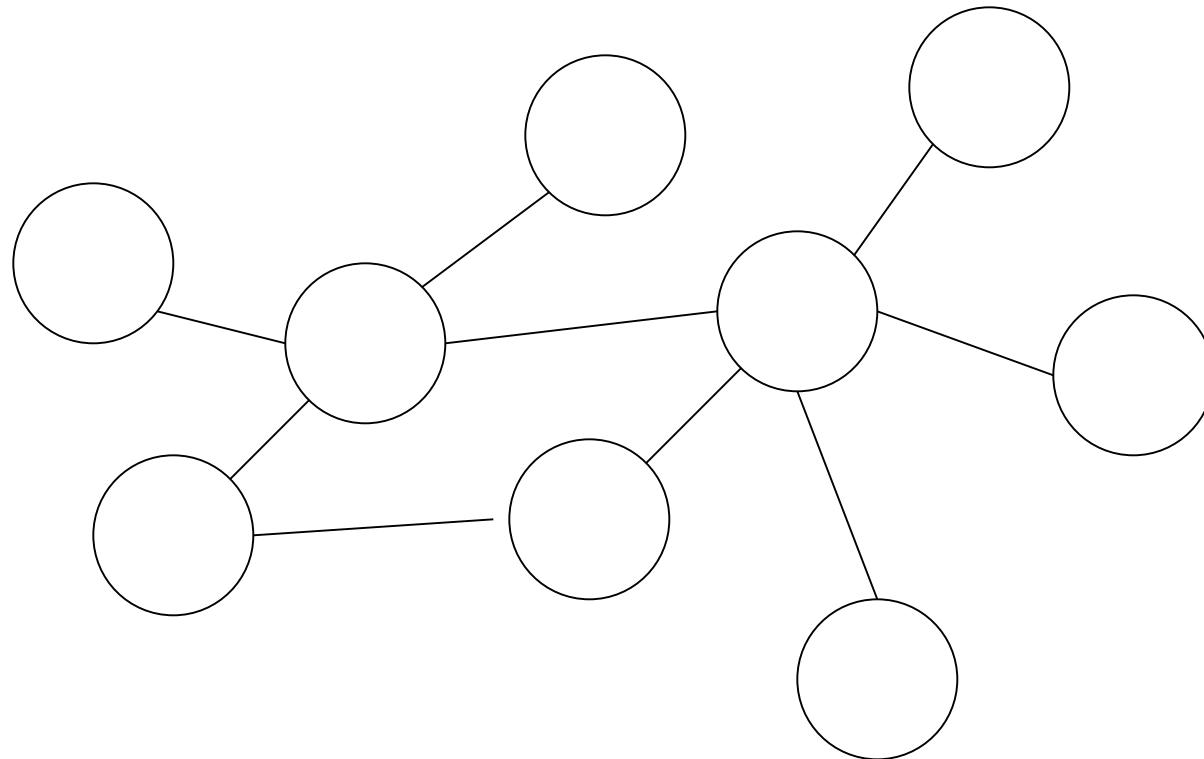
My e-mail is gite-vivek@usa.net.

Networking Fundamentals

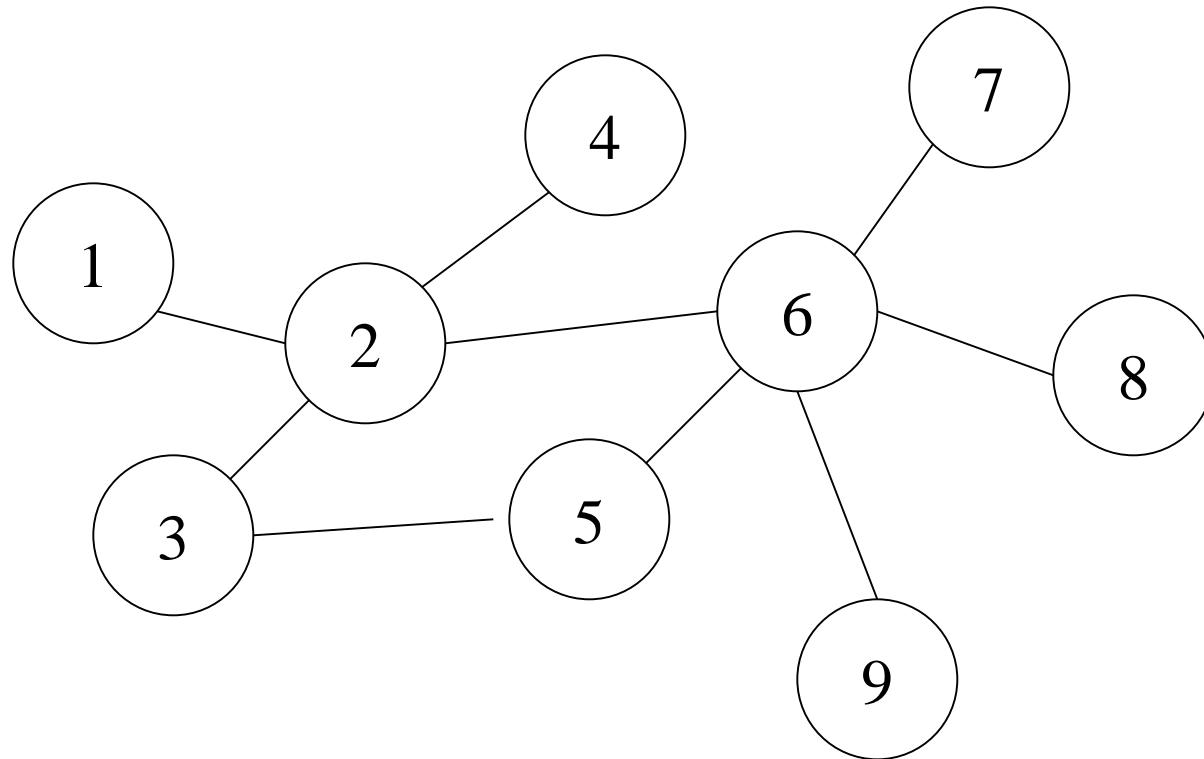
Basics

- Network – collection of nodes and links that cooperate for communication
- Nodes – computer systems
 - Internal (routers, bridges, switches)
 - Terminal (workstations)
- Links – connections for transmitting data
- Protocol – standards for formatting and interpreting data and control information

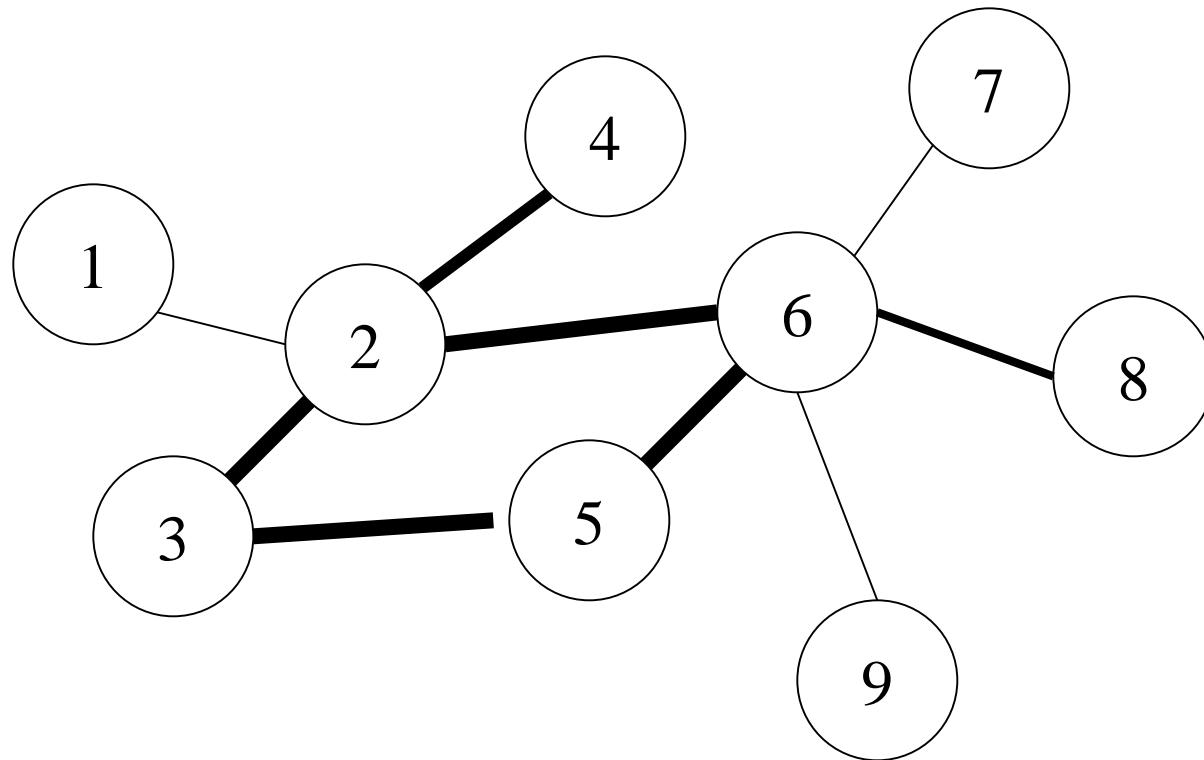
Network = {nodes and links}



Nodes have addresses



Links have bandwidths and latencies



Wires aren't perfect

- Attenuation (resistance)
 - degrades quality of signal
- Delay (speed of light * 2/3)
 - speed of light:
 - 8ms RTT coast-to-coast
 - 8 minutes to the sun
- Noise (microwaves and such)
- Nodes aren't perfect either
 - Unreliability is pervasive!

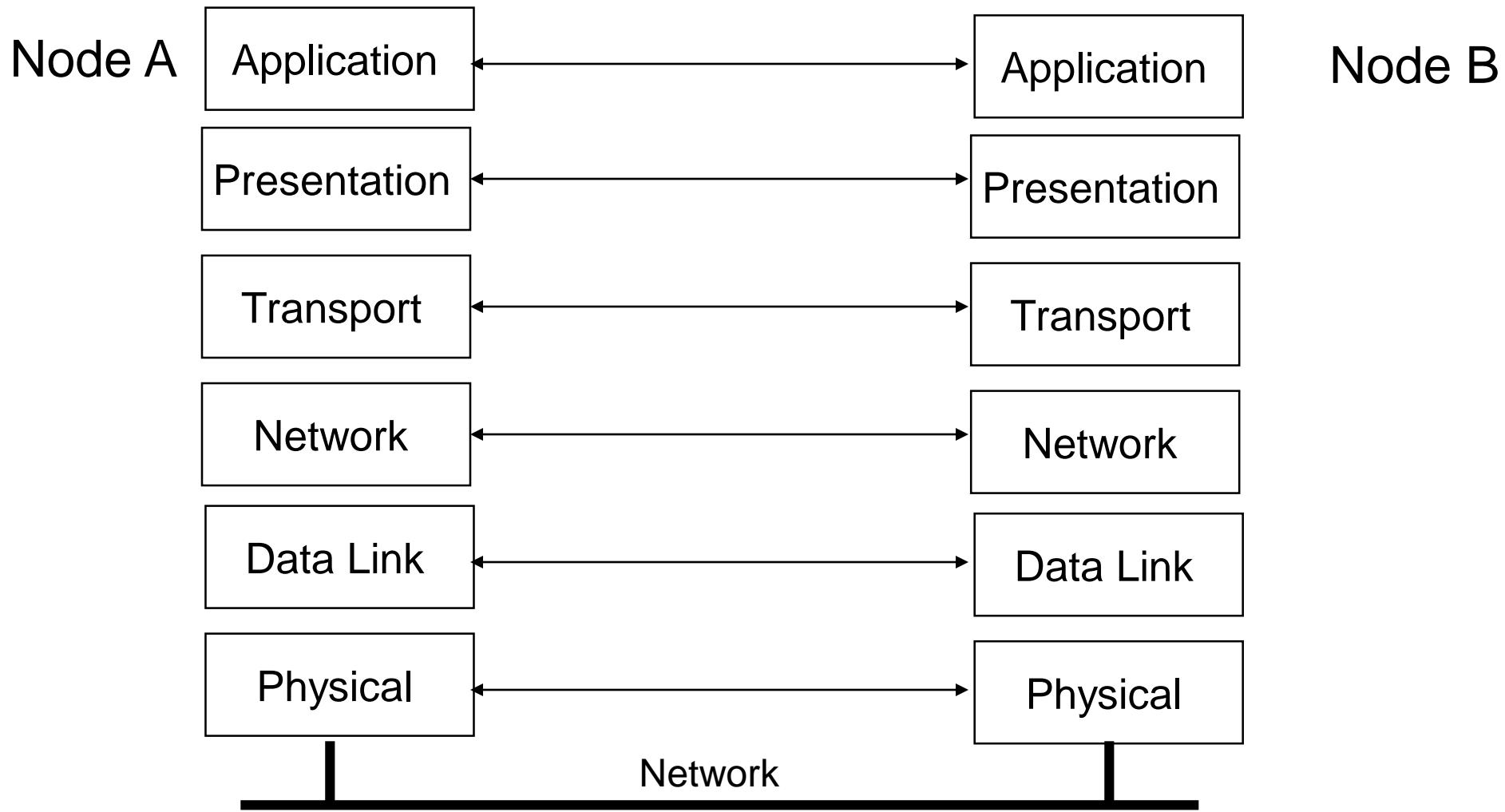
Getting Data Across (imperfect wires)

- Split up big files into small pieces
 - the pieces are called **packets**
- Each packet (~ 1500 bytes) is sent separately
 - packets can be corrupted
 - noise, bugs
 - packets can be dropped
 - corrupted, overloaded nodes
 - packets can be reordered
 - retransmission + different paths
- Allows packets from different flows to be multiplexed along the same link

Layers

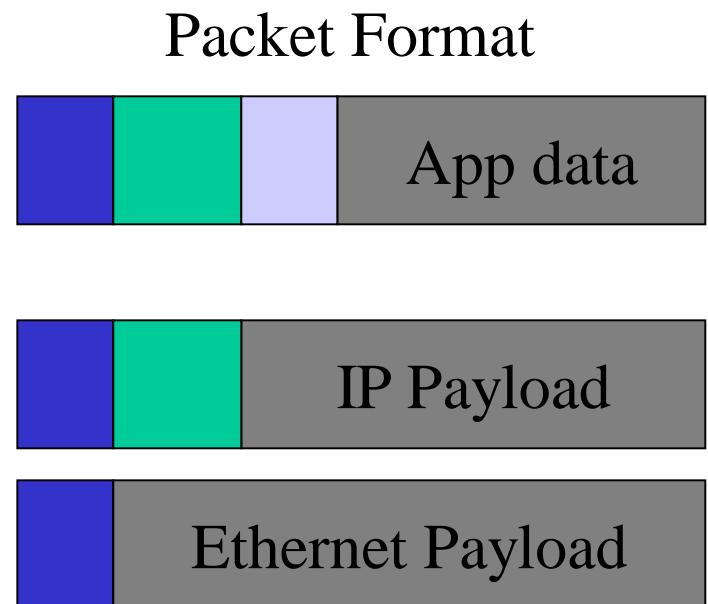
- Each layer abstracts the services of various lower layers, providing a uniform interface to higher layers.
- Each layer needs to know:
 - How to interpret a packet's payload
 - e.g., protocol numbers
 - How to use the services of a lower layer

OSI Levels



Layers

OSI Reference	Reality
Application	HTTP
Presentation	
Session	
Transport	TCP
Network	IP
Data-Link	Ethernet
Physical	Twisted Pair



The Internet Protocol (IP)

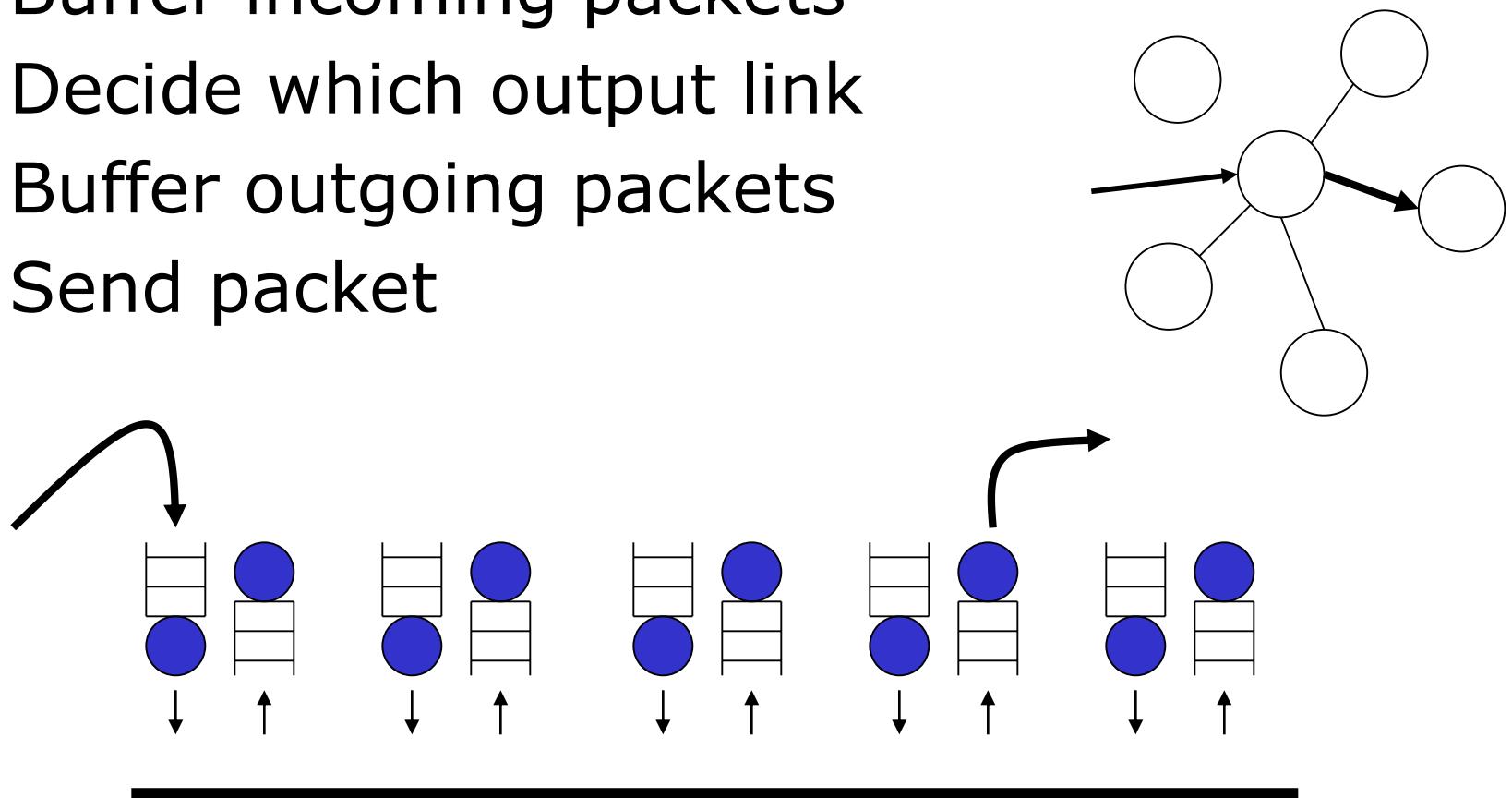
- Connects disparate networks
 - Single (hierarchical) address space
 - Single network header
- Assumes data link is unreliable
- Provides unreliable service
 - Loss: A B D E
 - Duplication: A B B C D E
 - Corruption: A Q C D E
 - Reordering: A C D B E

IP Addresses

- 32 bits long, split into 4 octets:
 - For example, 128.95.2.24
- Hierarchical:
 - First bits describe which network
 - Last bits describe which host on the network
- UW subnets include:
 - 128.95, 140.142...
- UW CSE subnets include:
 - 128.95.2, 128.95.4, 128.95.219...

Packet Forwarding

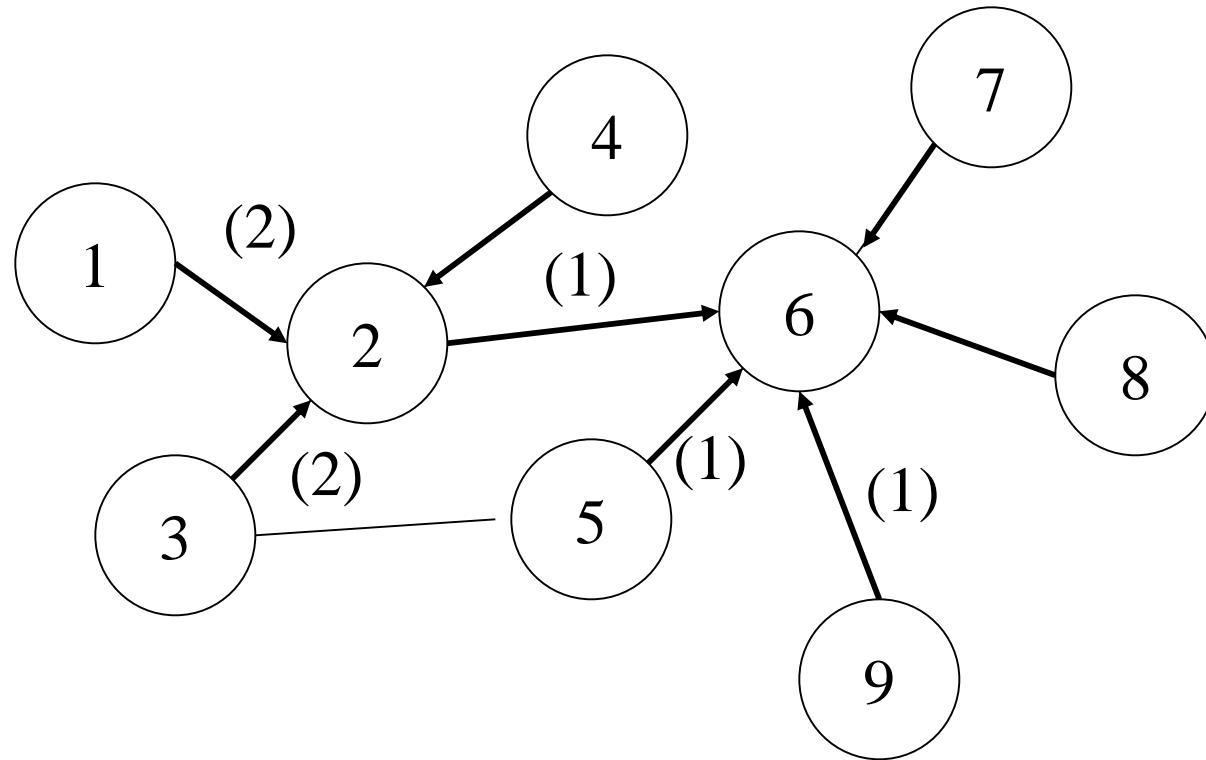
- Buffer incoming packets
- Decide which output link
- Buffer outgoing packets
- Send packet



Routing

- How do nodes determine which output link to use to reach a destination?
- Distributed algorithm for converging on shortest path tree
- Nodes exchange reachability information:
 - “I can get to 128.95.2.x in 3 hops”

Shortest path tree



(x) Is the cost to get to 6. The metric (cost per link) here is 1.
Simple algorithm: 6 broadcasts “I’m alive” to neighbors.
Neighbors send “I can get to 6 in 1 hop”, etc.

Route Aggregation

- What hierarchical addressing is good for.
- UW routers can advertise 128.95.x.x
 - instead of 128.95.2.x, 128.95.3,x, ...
- Other routers don't need forwarding table entries for each host in the network.

Routing Reality

- Routing in the Internet connects Autonomous Systems (AS's)
 - AT&T, Sprint, UUNet, BBN...
- Shortest path, sort of... money talks.
 - actually a horrible mess
 - nobody really knows what's going on
 - get high \$\$ job if you are a network engineer that messes with this stuff

TCP Service Model

- Provide reliability, ordering on the unreliable, unordered IP
- Bytestream oriented: when you send data using TCP, you think about bytes, not about packets.

TCP Ports

- Connections are identified by the tuple:
 - IP source address
 - IP destination address
 - TCP source port
 - TCP destination port
- Allows multiple connections; multiple application protocols, between the same machines
- Well known ports for some applications: (web: 80, telnet: 23, mail:25, dns: 53)

TCP's Sliding Window

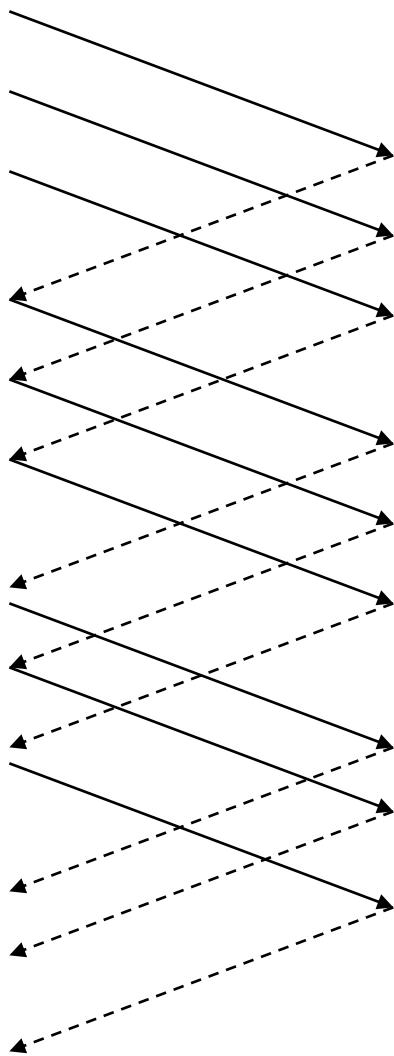
- Simple reliability:
 - Send one packet, wait for acknowledgment, then send the next...
- Better performance:
 - Keep several unacknowledged packets in the network (a window)

Sliding Window Example

Send 1
Send 2
Send 3

Send 4
Send 5
Send 6

Send 7
Send 8
Send 9



Ack 1
Ack 2
Ack 3

Ack 4
Ack 5
Ack 6

Ack 7
Ack 8

- Window size = 3
- Can send up to three packets into the network at a time.
- Each packet has a sequence number for ordering

TCP's Congestion Control

- How big should the window be?
- Performance is limited by:
 - (window size) / round trip time
 - Performance of bottleneck link (modem?)
- If window is too small, performance is wasted.
- If window is too big, may overflow network buffers, causing packet loss.

Steps for a web access

- Name lookup
 - Client to local DNS server
 - Local DNS may return a cached binding, or lookup the name for itself
- TCP Connection setup
 - Client to remote IP, port 80
- Send HTTP request
 - “GET /index.html”
- Receive HTTP response
 - “blah blah blah” maybe several packets
- TCP Connection teardown

HTTP 1.1

- Incremental improvements
- “Persistent connections” allow multiple requests over the same connection
 - Web transfers are often small
 - Avoid connection setup and teardown overhead
 - TCP is better the longer you use it: it learns how fast to send to get best performance without overflowing buffers.



UNC
INFORMATION
TECHNOLOGY SERVICES

Shell Scripting

Santosh Kumar
Assistant Professor
AI&DS Department, VIIT, Pune
santosh.kumar@viit.ac.in

- Introduction
 - UNIX/LINUX and Shell
 - UNIX Commands and Utilities
 - Basic Shell Scripting Structure
- Shell Programming
 - Variable
 - Operators
 - Logic Structures
- Examples of Application in Cloud & Devops
- Hands-on Exercises

Why Shell Scripting ?

- Shell scripts can be used to prepare input files, job monitoring, and output processing.
- Useful to create own commands.
- Save lots of time on file processing.
- To automate some task of day to day life.
- System Administration part can be also automated.

Objectives & Prerequisites

- **After this workshop, you should be:**
 - Familiar with UNIX/LINUX, Borne Shell, shell variables/operators
 - Able to write simple shell scripts to illustrate programming logic
 - Able to write scripts for Cloud computing purposes
- **We assume that you have/know**
 - An account on the Emerald cluster
 - Basic knowledge of UNIX/LINUX and commands
 - UNIX editor e.g. vi or emacs

History of UNIX/Linux

- Unix is a command line operating system developed around 1969 in the Bell Labs
- Originally written using C
- Unix is designed so that users can extend the functionality
 - To build new tools easily and efficiently
 - To customize the shell and user interface.
 - To string together a series of Unix commands to create new functionality.
 - To create custom commands that do exactly what we want.
- Around 1990 Linus Torvalds of Helsinki University started off a freely available academic version of Unix
- Linux is the Antidote to a Microsoft dominated future

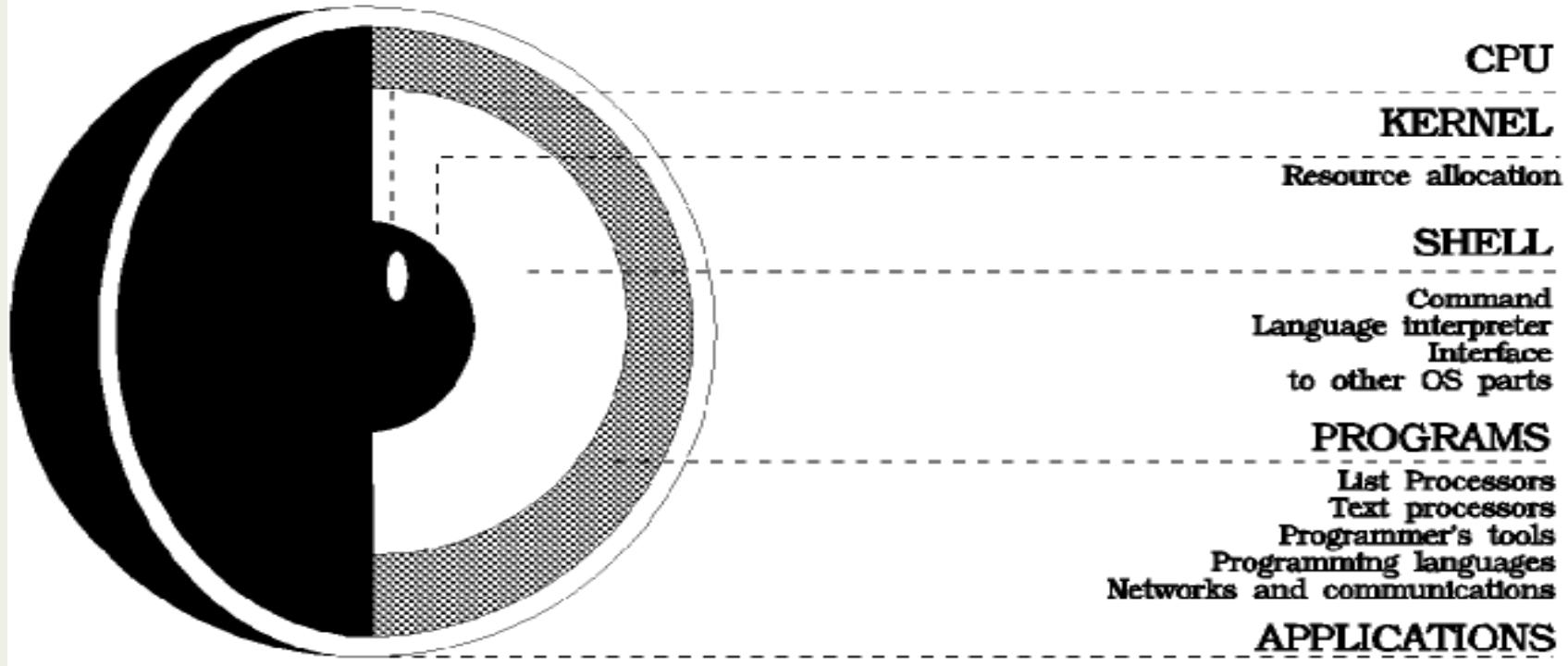
What is UNIX/Linux ?

Simply put

- Multi-Tasking O/S
- Multi-User O/S
- Available on a range of Computers

- | | |
|---------|------------------|
| ■ SunOS | Sun Microsystems |
| ■ IRIX | Silicon Graphics |
| ■ HP-UX | Hewlett Packard |
| ■ AIX | IBM |
| ■ Linux | |

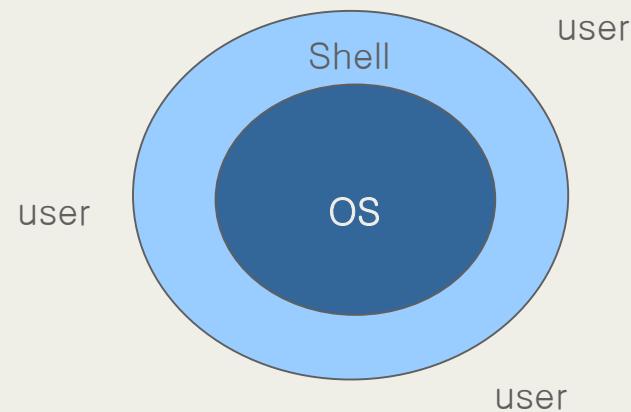
UNIX/LINUX Architecture



What is a “Shell”?

- The “Shell” is simply *another program* on top of the kernel which provides a basic human-OS interface.
 - It is a command interpreter
 - ◆ Built on top of the kernel
 - ◆ Enables users to run services provided by the UNIX OS
 - In its simplest form, a series of commands in a file is a shell program that saves having to retype commands to perform common tasks.
- How to know what shell you use

```
echo $SHELL
```



- sh Bourne Shell (Original Shell) (*Steven Bourne of AT&T*)
- bash Bourne Again Shell (*GNU Improved Bourne Shell*)
- csh C-Shell (C-like Syntax)(*Bill Joy of Univ. of California*)
- ksh Korn-Shell (Bourne+some C-shell)(*David Korn of AT&T*)
- tcsh Turbo C-Shell (More User Friendly C-Shell).
- To check shell:
 - \$ echo \$SHELL (shell is a pre-defined variable)
- To switch shell:
 - \$ exec shellname (e.g., \$ exec bash or simply type \$ bash)
 - You can switch from one shell to another by just typing the name of the shell. `exit` return you back to previous shell.

Which Shell to Use?

- **sh** (Bourne shell) was considered better for programming
- **csh** (C-Shell) was considered better for interactive work.
- **tcsh** and **korn** were improvements on c-shell and bourne shell respectively.
- **bash** is largely compatible with sh and also has many of the nice features of the other shells
- On many systems such as our LINUX clusters sh is symbolically linked to bash, /bin/sh -> /bin/bash
- We recommend that you use sh/bash for writing new shell scripts but learn csh/tcsh to understand existing scripts.
- Many, if not all, scientific applications require csh/tcsh environment (GUI, Graphics Utility Interface)
- **All Linux versions use the Bash shell (Bourne Again Shell) as the default shell**
 - Bash/Bourn/ksh/sh prompt: \$
- **All UNIX system include C shell and its predecessor Bourne shell.**
 - Csh/tcsh prompt: %



What is Shell Script?

- A **shell script** is a script written for the shell
- Two key ingredients
 - UNIX/LINUX commands
 - Shell programming syntax

A Shell Script Example

```
#!/bin/sh

`ls -l *.log| awk '{print $8}' |sed 's/.log//g' > file_list` 

cat file_list|while read each_file
do
    babel -ig03 $each_file".log" -oxyz $each_file".xyz"

    echo '# nosymmetry integral=Grid=UltraFine scf=tight rhf/6-311++g** pop=(nbo,chelpg)'>head
    echo '' >>head
    echo "$each_file" opt pop nbo chelp aim charges '>> head
    echo '' >>head
    echo '0 1 '>>head

    `sed '1,2d' $each_file.xyz >junk`
    input=./$each_file".com"
    cat head > $input
    cat junk >> $input
    echo '' >> $input

done
/bin/rm ./junk ./head ./file_list
```

UNIX/LINUX Commands

- File Management and Viewing
 - Filesystem Management
 - Help,Job/Process Management
 - Network Management
 - System Management
 - User Management
 - Printing and Programming
 - Document Preparation
 - Miscellaneous
- To understand the working of the command and possible options use ([man](#) command)
 - Using the GNU Info System ([info](#), info command)
 - Listing a Description of a Program ([whatis](#) command)
 - Many tools have a long-style option, `--help', that outputs usage information about the tool, including the options and arguments the tool takes. Ex: `whoami --help`

File and Directory Management

- **cd** Change the current directory. With no arguments "cd" changes to the users home directory. (cd <directory path>)
- **chmod** Change the file permissions.

Ex: chmod 751 myfile : change the file permissions to rwx for owner, rx for group and x for others (**x=1,r=4,w=2**)

Ex: chmod go=+r myfile : Add read permission for the group and others (character meanings u-user, g-group, o-other, + add permission,-remove,r-read,w-write,x-exe)

Ex: chmod +s myfile - Setuid bit on the file which allows the program to run with user or group privileges of the file.

- **chown** Change owner.

Ex: chown <owner1> <filename> : Change ownership of a file to owner1.

- **chgrp** Change group.

Ex: chgrp <group1> <filename> : Change group of a file to group1.

- **cp** Copy a file from one location to another.

Ex: cp file1 file2 : Copy file1 to file2; Ex: cp –R dir1 dir2 : Copy dir1 to dir2

File and Directory Management

■ **ls** List contents of a directory.

Ex: ls, ls -l , ls -al, ls -ld, ls -R

■ **mkdir** Make a directory.

Ex: mkdir <directory name> : Makes a directory

Ex *mkdir -p /www/chache/var/log* will create all the directories starting from www.

■ **mv** Move or rename a file or directory.

Ex: mv <source> <destination>

■ **find** Find files (find <start directory> -name <file name> -print)

Ex: *find /home -name readme -print*

Search for readme starting at home and output full path, "/home" = Search starting at the home directory and proceed through all its subdirectories; "-name readme" = Search for a file named readme "-print" = Output the full path to that file

■ **locate** File locating program that uses the slocate database.

Ex: locate -u to create the database,

locate <file/directory> to find file/directory

- **pwd** Print or list the present working directory with full path.
- **rm** Delete files (Remove files). (`rm -rf <directory/file>`)
- **rmdir** Remove a directory. The directory must be empty. (`rmdir <directory>`)
- **touch** Change file timestamps to the current time. Make the file if it doesn't exist. (`touch <filename>`)
- **whereis** Locate the binary and man page files for a command. (`whereis <program/command>`)
- **which** Show full path of commands where given commands reside. (`which <command>`)

File viewing and editing

- **emacs** Full screen editor.
- **pico** Simple text editor.
- **vi** Editor with a command mode and text mode. Starts in command mode.
- **gedit** GUI Text Editor
- **tail** Look at the last 10 lines of a file.

Ex: `tail -f <filename>` ; Ex: `tail -100 <filename>`

- **head** Look at the first 10 lines of a file. (`head <filename>`)

File compression, backing up and restoring

- **compress** Compress data.
- **uncompress** Expand data.
- **cpio** Can store files on tapes. to/from archives.
- **gzip** - zip a file to a gz file.
- **gunzip** - unzip a gz file.
- **tar** Archives files and directories. Can store files and directories on tapes.

Ex: tar -zcvf <destination> <files/directories> - Archive copy groups of files. tar –zxvf <compressed file> to uncompress

- **zip** – Compresses a file to a .zip file.
- **unzip** – Uncompresses a file with .zip extension.
- **cat** View a file

Ex: cat filename

- **cmp** Compare two files.
- **cut** Remove sections from each line of files.

File and Directory Management

- **diff** Show the differences between files.
Ex: `diff file1 file2` : Find differences between file1 & file2.
- **echo** Display a line of text.
- **grep** List all files with the specified expression.
(*grep pattern <filename/directorypath>*)
Ex: `ls -l |grep sidbi` : List all lines with a sidbi in them.
Ex: `grep " R "` : Search for R with a space on each side
- **sleep** Delay for a specified amount of time.
- **sort** Sort a file alphabetically.
- **uniq** Remove duplicate lines from a sorted file.
- **wc** Count lines, words, characters in a file. (`wc -c/w/l <filename>`).
- **sed** stream editor, extremely powerful!
- **awk** an extremely versatile programming language for working on files

Useful Commands in Scripting

- grep
 - Pattern searching
 - Example: `grep 'boo' filename`
- sed
 - Text editing
 - Example: `sed 's/XYZ/xyz/g' filename`
- awk
 - Pattern scanning and processing
 - Example: `awk '{print $4, $7}' filename`

Shell Scripting

- Start `vi scriptfilename.sh` with the line
`#!/bin/sh`
- All other lines starting with # are comments.
 - make code readable by including comments
- Tell Unix that the script file is executable
 - `$ chmod u+x scriptfilename.sh`
 - `$ chmod +x scriptfilename.sh`
- Execute the shell-script
 - `$./scriptfilename.sh`

My First Shell Script

```
$ vi myfirstscript.sh
```

```
#! /bin/sh
```

```
# The first example of a shell script
```

```
directory=`pwd`
```

```
echo Hello World!
```

```
echo The date today is `date`
```

```
echo The current directory is $directory
```

```
$ chmod +x myfirstscript.sh
```

```
$ ./myfirstscript.sh
```

```
Hello World!
```

```
The date today is Mon Mar 8 15:20:09 EST 2010
```

```
The current directory is /netscr/shubin/test
```

Shell Scripts

- **Text files that contain sequences of UNIX commands , created by a text editor**
- **No compiler required to run a shell script, because the UNIX shell acts as an **interpreter** when reading script files**
- **After you create a shell script, you simply tell the OS that the file is a program that can be executed, by using the **chmod** command to change the files' mode to be executable**
- **Shell programs run **less quickly** than compiled programs, because the shell must interpret each UNIX command inside the executable script file before it is executed**

Commenting

- Lines starting with # are comments except the very first line where # ! indicates the location of the shell that will be run to execute the script.
- On any line characters following an unquoted # are considered to be comments and ignored.
- Comments are used to;
 - Identify who wrote it and when
 - Identify input variables
 - Make code easy to read
 - Explain complex code sections
 - Version control tracking
 - Record modifications

Quote Characters

There are three different quote characters with different behaviour. These are:

- “ : **double quote**, weak quote. If a string is enclosed in “ ” the references to variables (i.e `$variable`) are replaced by their values. Also back-quote and escape \ characters are treated specially.
- ‘ : **single quote**, strong quote. Everything inside single quotes are taken literally, nothing is treated as special.
- ` : **back quote**. A string enclosed as such is treated as a command and the shell attempts to execute it. If the execution is successful the primary output from the command replaces the string.

Example: `echo "Today is:" `date``

Echo command is well appreciated when trying to debug scripts.

Syntax : echo {options} string

Options: -e : expand \ (back-slash) special characters

-n : do not output a new-line at the end.

String can be a “weakly quoted” or a ‘strongly quoted’ string. In the weakly quoted strings the references to variables are replaced by the value of those variables before the output.

As well as the variables some special backslash_escaped symbols are expanded during the output. If such expansions are required the -e option must be used.

User Input During Shell Script Execution

- As shown on the hello script input from the standard input location is done via the read command.
- Example

```
echo "Please enter three filenames:"  
read filea fileb filec  
echo "These files are used:$filea $fileb $filec"
```

- Each read statement reads an entire line. In the above example if there are less than 3 items in the response the trailing variables will be set to blank ‘ ’.
- Three items are separated by one space.

Hello script exercise continued...

- The following script asks the user to enter his name and displays a personalised hello.

```
#!/bin/sh  
  
echo "Who am I talking to?"  
  
read user_name  
  
echo "Hello $user_name"
```

- Try replacing “ with ‘ in the last line to see what happens.

Debugging your shell scripts

- Generous use of the `echo` command will help.
- Run script with the `-x` parameter.
E.g. `sh -x ./myscript`
or `set -o xtrace` before running the script.
- These options can be added to the first line of the script where the shell is defined.
e.g. `#!/bin/sh -xv`

Shell Programming

- **Programming features of the UNIX/LINUX shell:**
 - **Shell variables:** Your scripts often need to keep values in memory for later use. Shell variables are symbolic names that can access values stored in memory
 - **Operators:** Shell scripts support many operators, including those for performing mathematical operations
 - **Logic structures:** Shell scripts support **sequential logic** (for performing a series of commands), **decision logic** (for branching from one point in a script to another), **looping logic** (for repeating a command several times), and **case logic** (for choosing an action from several possible alternatives)

- **Variables are symbolic names that represent values stored in memory**
- **Three different types of variables**
 - **Global Variables:** Environment and configuration variables, capitalized, such as **HOME, PATH, SHELL, USERNAME, and PWD.**

When you login, there will be a large number of global System variables that are already defined. These can be freely referenced and used in your shell scripts.

- **Local Variables**

Within a shell script, you can create as many new variables as needed. Any variable created in this manner remains in existence only within that shell.

- **Special Variables**

Reversed for OS, shell programming, etc. such as positional parameters \$0, \$1 ...

A few global (environment) variables

SHELL	Current shell
DISPLAY	Used by X-Windows system to identify the display
HOME	Fully qualified name of your login directory
PATH	Search path for commands
MANPATH	Search path for <man> pages
PS1 & PS2	Primary and Secondary prompt strings
USER	Your login name
TERM	terminal type
PWD	Current working directory

Referencing Variables

Variable contents are accessed using '\$':

e.g. **\$ echo \$HOME**

\$ echo \$SHELL

To see a list of your environment variables:

\$ printenv

or:

\$ printenv | more

Defining Local Variables

- As in any other programming language, variables can be defined and used in shell scripts.
- Unlike other programming languages, variables in Shell Scripts are not typed.
- Examples :

a=1234 # a is NOT an integer, a string instead

b=\$a+1 # will not perform arithmetic but be the string '1234+1'

b=`expr \$a + 1` will perform arithmetic so b is 1235 now.

Note : +,-,/,*,**, % operators are available.

b=abcde # b is string

b='abcde' # same as above but much safer.

b=abc def # will not work unless 'quoted'

b='abc def' # i.e. this will work.

IMPORTANT NOTE: DO NOT LEAVE SPACES AROUND THE =

Referencing variables --curly bracket

- Having defined a variable, its contents can be referenced by the \$ symbol. E.g. \${variable} or simply \$variable. When ambiguity exists \$variable will not work. Use \${ } the rigorous form to be on the safe side.
- Example:

```
a='abc'
```

```
b=${a}def # this would not have worked without the{ } as  
#it would try to access a variable named adef
```

- To create lists (array) – round bracket
\$ set Y = (UNL 123 CS251)
- To set a list element – square bracket
\$ set Y[2] = HUSKER
- To view a list element:
\$ echo \$Y[2]
- Example:

```
#!/bin/sh
a=(1 2 3)
echo ${a[*]}
echo ${a[0]}
```

Results: 1 2 3

1



Positional Parameters

- When a shell script is invoked with a set of command line parameters each of these parameters are copied into special variables that can be accessed.
- **\$0** This variable that contains the name of the script
- **\$1, \$2, \$n** 1st, 2nd 3rd command line parameter
- **\$#** Number of command line parameters
- **\$\$** process ID of the shell
- **\$@** same as **\$*** but as a list one at a time (see for loops later)
- **\$?** Return code ‘exit code’ of the last command
- **Shift** command: This shell command shifts the positional parameters by one towards the beginning and drops \$1 from the list. After a shift \$2 becomes \$1 , and so on ... It is a useful command for processing the input parameters one at a time.

Example:

Invoke : ./myscript one two buckle my shoe

During the execution of **myscript** variables **\$1 \$2 \$3 \$4** and **\$5** will contain the values **one, two, buckle, my, shoe** respectively.

- **vi myinputs.sh**

```
#!/bin/sh
echo Total number of inputs: $#
echo First input: $1
echo Second input: $2
```
- **chmod u+x myinputs.sh**
- **myinputs.sh HUSKER UNL CSE**
Total number of inputs: 3
First input: HUSKER
Second input: UNL

- programming features of the UNIX shell:
 - ***Shell variables***
 - ***Operators***
 - ***Logic structures***

Shell Operators

- The Bash/Bourne/ksh shell operators are divided into three groups: **defining and evaluating operators**, **arithmetic operators**, and **redirecting and piping operators**

Defining and Evaluating

- A shell variable take on the generalized form **variable=value** (except in the C shell).

```
$ set x=37; echo $x
```

37

```
$ unset x; echo $x
```

x: Undefined variable.

- You can set a pathname or a command to a variable or substitute to set the variable.

```
$ set mydir=`pwd`; echo $mydir
```

Pipes & Redirecting

- **Piping:** An important early development in Unix , a way to pass the output of one tool to the input of another.

```
$ who | wc -l
```

By combining these two tools, giving the wc command the output of who, you can build a new command to **list the number of users currently on the system**

- **Redirecting via angle brackets:** Redirecting input and output follows a similar principle to that of piping except that redirects work with files, not commands.

```
tr '[a-z]' '[A-Z]' < $in_file > $out_file
```

The command must come first, the *in_file* is directed in by the less_than sign (<) and the *out_file* is pointed at by the greater_than sign (>).



Arithmetic Operators

- **expr supports the following operators:**
 - arithmetic operators: +,-,*/,%
 - comparison operators: <, <=, ==, !=, >=, >
 - boolean/logical operators: &, |
 - parentheses: (,)
 - precedence is the same as C, Java



- **vi math.sh**

```
#!/bin/sh
count=5
count=`expr $count + 1`
echo $count
```

- **chmod u+x math.sh**
- **math.sh**

- **vi real.sh**

```
#!/bin/sh
a=5.48
b=10.32
c=`echo "scale=2; $a + $b" |bc`
echo $c
```

- **chmod u+x real.sh**

- **./real.sh**

15.80

Arithmetic operations in shell scripts

<code>var++ , var-- , ++var , --var</code>	post/pre increment/decrement
<code>+ , -</code>	add subtract
<code>* , / , %</code>	multiply/divide, remainder
<code>**</code>	power of
<code>! , ~</code>	logical/bitwise negation
<code>& , </code>	bitwise AND, OR
<code>&& </code>	logical AND, OR

- programming features of the UNIX shell:
 - ***Shell variables***
 - ***Operators***
 - ***Logic structures***

Shell Logic Structures

The four basic logic structures needed for program development are:

- **Sequential logic:** to execute commands in the order in which they appear in the program
- **Decision logic:** to execute commands only if a certain condition is satisfied
- **Looping logic:** to repeat a series of commands for a given number of times
- **Case logic:** to replace “if then/else if/else” statements when making numerous comparisons

Conditional Statements (if constructs)

The most general form of the if construct is;

```
if command executes successfully
then
    execute command
elif this command executes successfully
then
    execute this command
    and execute this command
else
    execute default command
fi
```

However- elif and/or else clause can be omitted.

Examples

SIMPLE EXAMPLE:

```
if date | grep "Fri"
then
    echo "It's Friday!"
fi
```

FULL EXAMPLE:

```
if [ "$1" == "Monday" ]
then
    echo "The typed argument is Monday."
elif [ "$1" == "Tuesday" ]
then
    echo "Typed argument is Tuesday"
else
    echo "Typed argument is neither Monday nor Tuesday"
fi
```

Note: = or == will both work in the test but == is better for readability.

String and numeric comparisons used with test or [[]] which is an alias for test and also [] which is another acceptable syntax

- `string1 = string2` True if strings are identical
...ditto....
- `string1 !=string2` True if strings are not identical
- `string` Return 0 exit status (=true) if string is not null
- `-n string` Return 0 exit status (=true) if string is not null
- `-z string` Return 0 exit status (=true) if string is null

- `int1 -eq int2` Test identity
- `int1 -ne int2` Test inequality
- `int1 -lt int2` Less than
- `int1 -gt int2` Greater than
- `int1 -le int2` Less than or equal
- `int1 -ge int2` Greater than or equal

Combining tests with logical operators || (or) and && (and)

Syntax: if cond1 && cond2 || cond3 ...

An alternative form is to use a compound statement using the -a and -o keywords, i.e.

```
if cond1 -a cond2 -o cond3 ...
```

Where cond1,2,3 .. Are either commands returning a value or test conditions of the form [] or test ...

Examples:

```
if date | grep "Fri" && `date +'%H'` -gt 17
```

```
then
```

```
    echo "It's Friday, it's home time!!!"
```

```
fi
```

```
if [ "$a" -lt 0 -o "$a" -gt 100 ] # note the spaces around ] and [
```

```
then
```

```
    echo " limits exceeded"
```

```
fi
```

File enquiry operations

-d file	Test if file is a directory
-f file	Test if file is not a directory
-s file	Test if the file has non zero length
-r file	Test if the file is readable
-w file	Test if the file is writable
-x file	Test if the file is executable
-o file	Test if the file is owned by the user
-e file	Test if the file exists
-z file	Test if the file has zero length

All these conditions return true if satisfied and false otherwise.

■ A simple example

```
#!/bin/sh

if [ "$#" -ne 2 ] then
    echo $0 needs two parameters!
    echo You are inputting $# parameters.

else
    par1=$1
    par2=$2

fi

echo $par1
echo $par2
```

Another example:

```
#! /bin/sh
# number is positive, zero or negative
echo -e "enter a number:\c"
read number
if [ "$number" -lt 0 ]
then
    echo "negative"
elif [ "$number" -eq 0 ]
then
    echo zero
else
    echo positive
fi
```

Loop is a block of code that is repeated a number of times.

The repeating is performed either a pre-determined number of times determined by a list of items in the loop count (**for loops**) or until a particular condition is satisfied (**while** and **until loops**)

To provide flexibility to the loop constructs there are also two statements namely **break** and **continue** are provided.

for loops

Syntax:

```
for arg in list
do
    command(s)
...
done
```

Where the value of the variable *arg* is set to the values provided in the list one at a time and the block of statements executed. This is repeated until the list is exhausted.

Example:

```
for i in 3 2 5 7
do
    echo "$i times 5 is $(( i * 5 )) "
done
```

The while Loop

- A different pattern for looping is created using the **while statement**
- The **while statement** best illustrates how to set up a loop to test repeatedly for a matching condition
- The while loop tests an expression in a manner similar to the if statement
- As long as the statement inside the brackets is true, the statements inside the do and done statements repeat

while loops

Syntax:

```
while this_command_execute_successfully
do
    this command
    and this command
done
```

EXAMPLE:

```
while test "$i" -gt 0      # can also be while [ $i > 0 ]
do
    i=`expr $i - 1`
done
```

Looping Logic

- Example:
- Adding integers from 1 to 10

```
#!/bin/sh
for person in Bob Susan Joe Gerry
do
    echo Hello $person
done
```

Output:

```
Hello Bob
Hello Susan
Hello Joe
Hello Gerry
```

```
#!/bin/sh
i=1
sum=0
while [ "$i" -le 10 ]
do
    echo Adding $i into the sum.
    sum=`expr $sum + $i `
    i=`expr $i + 1 `
done
echo The sum is $sum.
```

until loops

The syntax and usage is almost identical to the while-loops.

Except that the block is executed until the test condition is satisfied, which is the opposite of the effect of test condition in while loops.

Note: You can think of *until* as equivalent to *not_while*

Syntax:

```
until test
do
  commands ....
done
```

Switch/Case Logic

- The **switch logic structure simplifies the selection of a match when you have a list of choices**
- It **allows your program to perform one of many actions, depending upon the value of a variable**

Case statements

The case structure compares a string ‘usually contained in a variable’ to one or more patterns and executes a block of code associated with the matching pattern. Matching-tests start with the first pattern and the subsequent patterns are tested only if no match is not found so far.

case argument in

pattern 1) execute this command

and this

and this;;

pattern 2) execute this command

and this

and this;;

esac

Functions

- Functions are a way of grouping together commands so that they can later be executed via a single reference to their name. If the same set of instructions have to be repeated in more than one part of the code, this will save a lot of coding and also reduce possibility of typing errors.

SYNTAX:

```
functionname()  
{  
    block of commands  
}  
#!/bin/sh  
  
    sum() {  
        x=`expr $1 + $2`  
        echo $x  
    }
```

```
sum 5 3
```

```
echo "The sum of 4 and 7 is `sum 4 7`"
```

Take-Home Message

- **Shell script is a high-level language that must be converted into a low-level (machine) language by UNIX Shell before the computer can execute it**
- **UNIX shell scripts, created with the vi or other text editor, contain two key ingredients: a selection of UNIX commands glued together by Shell programming syntax**
- **UNIX/Linux shells are derived from the UNIX Bourne, Korn, and C/TCSH shells**
- **UNIX keeps three types of variables:**
 - Configuration; environmental; local
- **The shell supports numerous operators, including many for performing arithmetic operations**
- **The logic structures supported by the shell are sequential, decision, looping, and case**

To Script or Not to Script

■ Pros

- File processing
- Glue together compelling, customized testing utilities
- Create powerful, tailor-made manufacturing tools
- Cross-platform support
- Custom testing and debugging

■ Cons

- Performance slowdown
- Accurate scientific computing

Shell Scripting Examples

- Input file preparation
- Job submission
- Job monitoring
- Results processing

Input file preparation

```
#!/bin/sh

`ls -l *.log| awk '{print $8}' |sed 's/.log//g' > file_list` 

cat file_list|while read each_file
do
    babel -ig03 $each_file".log" -oxyz $each_file".xyz"

    echo '# nosymmetry integral=Grid=UltraFine scf=tight rhf/6-311++g** pop=(nbo,chelpg)'>head
    echo '' >>head
    echo "$each_file" opt pop nbo chelp aim charges '>> head
    echo '' >>head
    echo '0 1 '>>head

    `sed '1,2d' $each_file.xyz >junk`
    input=./$each_file".com"
    cat head > $input
    cat junk >> $input
    echo '' >> $input

done
/bin/rm ./junk ./head ./file_list
```

LSF Job Submission

```
$ vi submission.sh
```

```
#!/bin/sh -f
```

```
#BSUB -q week
```

```
#BSUB -n 4
```

```
#BSUB -o output
```

```
#BSUB -J job_type
```

```
#BSUB -R "RH5 span[ptile=4]"
```

```
#BSUB -a mpichp4
```

```
mpirun.lsf ./executable.exe
```

```
exit
```

```
$chmod +x submission.sh
```

```
$bsub < submission.sh
```

Results Processing

```

#!/bin/sh
`ls -l *.out| awk '{print $8}'|sed 's/.out//g' > file_list`
cat file_list|while read each_file
do
    file1=./$each_file".out"
    Ts=`grep 'Kinetic energy =' $file1 |tail -n 1|awk '{print $4}' `
    Tw=`grep 'Total Steric Energy:' $file1 |tail -n 1|awk '{print $4}' `
    TsVne=`grep 'One electron energy =' $file1 |tail -n 1|awk '{print $5}' `
    Vnn=`grep 'Nuclear repulsion energy' $file1 |tail -n 1|awk '{print $5}' `
    J=`grep 'Coulomb energy =' $file1 |tail -n 1|awk '{print $4}' `
    Ex=`grep 'Exchange energy =' $file1 |tail -n 1|awk '{print $4}' `
    Ec=`grep 'Correlation energy =' $file1 |tail -n 1|awk '{print $4}' `
    Etot=`grep 'Total DFT energy =' $file1 |tail -n 1|awk '{print $5}' `
    HOMO=`grep 'Vector' $file1 | grep 'Occ=2.00'|tail -n 1|cut -c35-47|sed 's/D/E/g' `
    orb=`grep 'Vector' $file1 | grep 'Occ=2.00'|tail -n 1|awk '{print $2}' `
    orb=`expr $orb + 1 `
    LUMO=`grep 'Vector' $file1 |grep 'Occ=0.00'|grep '$orb' |tail -n 1|cut -c35-47|sed 's/D/E/g' `
    echo $each_file $Etot $Ts $Tw $TsVne $J $Vnn $Ex $Ec $HOMO $LUMO $steric >>out
done
/bin/rm file_list

```

Reference Books



- **Class Shell Scripting**
<http://oreilly.com/catalog/9780596005955/>
- **LINUX Shell Scripting With Bash**
<http://ebooks.ebookmall.com/title/linux-shell-scripting-with-bash-burtsch-ebooks.htm>
- **Shell Script in C Shell**
<http://www.grymoire.com/Unix/CshTop10.txt>
- **Linux Shell Scripting Tutorial**
<http://www.freeos.com/guides/lsst/>
- **Bash Shell Programming in Linux**
http://www.arachnoid.com/linux/shell_programming.html
- **Advanced Bash-Scripting Guide**
<http://tldp.org/LDP/abs/html/>
- **Unix Shell Programming**
<http://ebooks.ebookmall.com/title/unix-shell-programming-kochan-wood-ebooks.htm>



Questions & Comments

Please direct comments/questions about research computing to

E-mail: research@unc.edu

Please direct comments/questions pertaining to this presentation to

E-Mail: shubin@email.unc.edu

The PPT file of this presentation is available here:

http://its2.unc.edu/divisions/rc/training/scientific/short_courses/Shell_Scripting.ppt



Hands-on Exercises

1. The simplest Hello World shell script - Echo command
2. Summation of two integers - If block
3. Summation of two real numbers - bc (basic calculator) command
4. Script to find out the biggest number in 3 numbers - If -elif block
5. Operation (summation, subtraction, multiplication and division) of two numbers - Switch
6. Script to reverse a given number - While block
7. A more complicated greeting shell script
8. Sort the given five numbers in ascending order (using array) - Do loop and array
9. Calculating average of given numbers on command line arguments - Do loop
10. Calculating factorial of a given number - While block
11. An application in research computing - Combining all above
12. **Optional:** Write own shell scripts for your own purposes if time permits

The PPT/WORD format of this presentation is available here:

<http://its2.unc.edu/divisions/rc/training/scientific/>

/afs/isis/depts/its/public_html/divisions/rc/training/scientific/short_courses/ 72

Unit-II AWS

Cloud Computing and DevOps

Ms. Vidya S. Gaikwad
vidya.gaikwad@viit.ac.in

Department of Computer Engineering



BRACT's, Vishwakarma Institute of Information Technology, Pune-48

**(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)**

Contents

- Cloud Fundamentals:
 - What is Cloud?
 - Properties of cloud
 - Benefits of using cloud,
 - Service models
 - Deployment models
- AWS Accounts –
 - The Basics, Creating an AWS Account
 - Securing An AWS Account
 - Creating a Budget
 - IAM
 - Creating Access keys and setting up AWS CLI v2 tools
 - Simple Storage Service (S3)

Contents

- VIRTUAL PRIVATE CLOUD (VPC) BASICS
- ELASTIC COMPUTE CLOUD (EC2) BASICS
- Infrastructure as Code (CloudFormation)
- Load Balancing
- Systems Manager
- AWS Lambda

- Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing.
- Instead of buying, owning, and maintaining physical data centres and servers, we can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider.

- Who is using cloud computing?
- Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications.
- For example, healthcare companies are using the cloud to develop more personalized treatments for patients.
- Financial services companies are using the cloud to power real-time fraud detection and prevention.
- Video game makers are using the cloud to deliver online games to millions of players around the world.

Benefits of Cloud

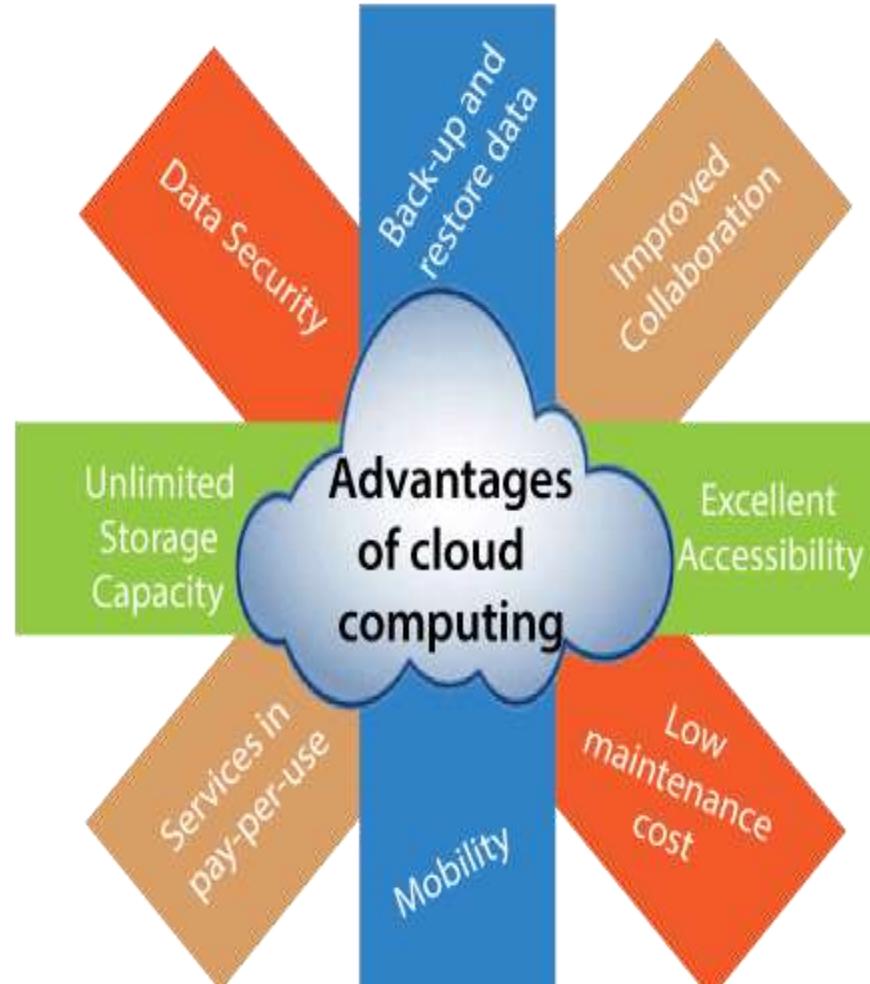


Figure 1: Benefits of Cloud [1]

1) Back-up and restore data

Once the data is stored in the cloud, it is easier to get back-up and restore that data using the cloud.

2) Improved collaboration

Cloud applications improve collaboration by allowing groups of people to quickly and easily share information in the cloud via shared storage.

3) Excellent accessibility

Cloud allows us to quickly and easily access store information anywhere, anytime in the whole world, using an internet connection. An internet cloud infrastructure increases organization productivity and efficiency by ensuring that our data is always accessible.

4) Low maintenance cost

Cloud computing reduces both hardware and software maintenance costs for organizations.

5) Mobility

Cloud computing allows us to easily access all cloud data via mobile.

6) Services in the pay-per-use model

Cloud computing offers Application Programming Interfaces (APIs) to the users for access services on the cloud and pays the charges as per the usage of service.

7) Unlimited storage capacity

Cloud offers us a huge amount of storing capacity for storing our important data such as documents, images, audio, video, etc. in one place.

8) Data security

Data security is one of the biggest advantages of cloud computing. Cloud offers many advanced features related to security and ensures that data is securely stored and handled.

What is AWS?

- AWS stands for **Amazon Web Services**.
- The AWS service is provided by the Amazon that uses distributed IT infrastructure to provide different IT resources available on demand.
- Provides different services such as infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS).

Pay-As-You-Go

Based on the concept of Pay-As-You-Go, AWS provides the services to the customers.

What is AWS?

- AWS provides services to customers when required without any prior commitment or upfront investment.
- Pay-As-You-Go enables the customers to procure services from AWS.
- Computing
- Programming models
- Database storage
- Networking



How to SignUp to the AWS platform

- Firstly visit the website <https://aws.amazon.com>.
- The following screen appears after opening the website, then click on the **Complete Sign Up** to create an account and fill the required details.



How to SignUp to the AWS platform

- The following screen appears after clicking on the "**Complete Sign Up**" button. If you are an already existing user of an AWS account, then enter the email address of your AWS account otherwise "**create an aws account**".
- On clicking on the "**create an aws account**" button, the following screen appears that requires some fields to be filled by the user.

The screenshot shows a web browser window with the URL <https://portal.aws.amazon.com/billing/signup#/start>. The title bar of the browser says "Create an AWS account". The page content includes a promotional message about free tier access and a form for creating a new AWS account. The form fields are:

Email address	<input type="text" value="gakshita123@gmail.com"/>
Password	<input type="password"/>
Confirm password	<input type="password"/>
AWS account name	<input type="text" value="Akshita Gupta"/>

Below the form is a yellow "Continue" button and a link to "Sign in to an existing AWS account". At the bottom, there is a copyright notice for Amazon Web Services, Inc. and links to "Privacy Policy" and "Terms of Use".

How to SignUp to the AWS platform

- Now, fill your contact information.

The screenshot shows the 'Contact Information' step of the AWS sign-up process. At the top right, there's a language selection dropdown set to 'English'. Below it, a note says 'All fields are required.' The form asks for account type (Professional or Personal), full name (Akhila Gupta), phone number (7842176966), country/region (India), address (sector -14 block c), city (Faridabad), state/province (Haryana), and postal code (121006). A note states: '* If you select India, your country/region selection cannot be changed after creating the account.' Below the address fields, there's a section about the Amazon Internet Services Pvt. Ltd. Customer Agreement, mentioning AISPL and its role as the local seller for AWS services in India. A checkbox is present for accepting the terms of the agreement. At the bottom is a yellow 'Create Account and Continue' button.

Contact Information

All fields are required.

Please select the account type and complete the fields below with your contact details.

Account type Professional Personal

Full name

Phone number

Country/Region

* If you select India, your country/region selection cannot be changed after creating the account.

Address
Apartment, suite, unit, building, floor, etc.

City

State / Province or region

Postal code

Amazon Internet Services Pvt. Ltd. Customer Agreement
Customers with an India contact address are now required to contract with Amazon Internet Services Private Ltd. (AISPL). AISPL is the local seller for AWS infrastructure services in India.

Check here to indicate that you have read and agree to the terms of the [AISPL Customer Agreement](#).

Create Account and Continue

© 2012 Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
[Privacy Policy](#) | [Terms of Use](#) | [Sign Out](#)

How to SignUp to the AWS platform

- After providing the contact information, fill your payment information.

The screenshot shows the 'Payment Information' step of the AWS sign-up process. At the top, the AWS logo is visible, and the text 'Payment Information' is centered. Below this, a note states: 'Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the [AWS Free Tier Limits](#). Review [frequently asked questions](#) for more information.' A callout box provides additional information about a \$2 charge during verification, which is refunded after validation. The form includes fields for 'Credit/Debit card number' (with an input field containing '|'), 'Expiration date' (with dropdown menus for month '01' and year '2019'), and 'Cardholder's name' (with an input field).

Payment Information

Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the [AWS Free Tier Limits](#). Review [frequently asked questions](#) for more information.

(i) As part of our card verification process we will charge INR 2 on your card when you click the "Secure Submit" button below. This will be refunded once your card has been validated. Your bank may take 3-5 business days to show the refund. Mastercard/Visa customers may be redirected to your bank website to authorize the charge.

Credit/Debit card number

|

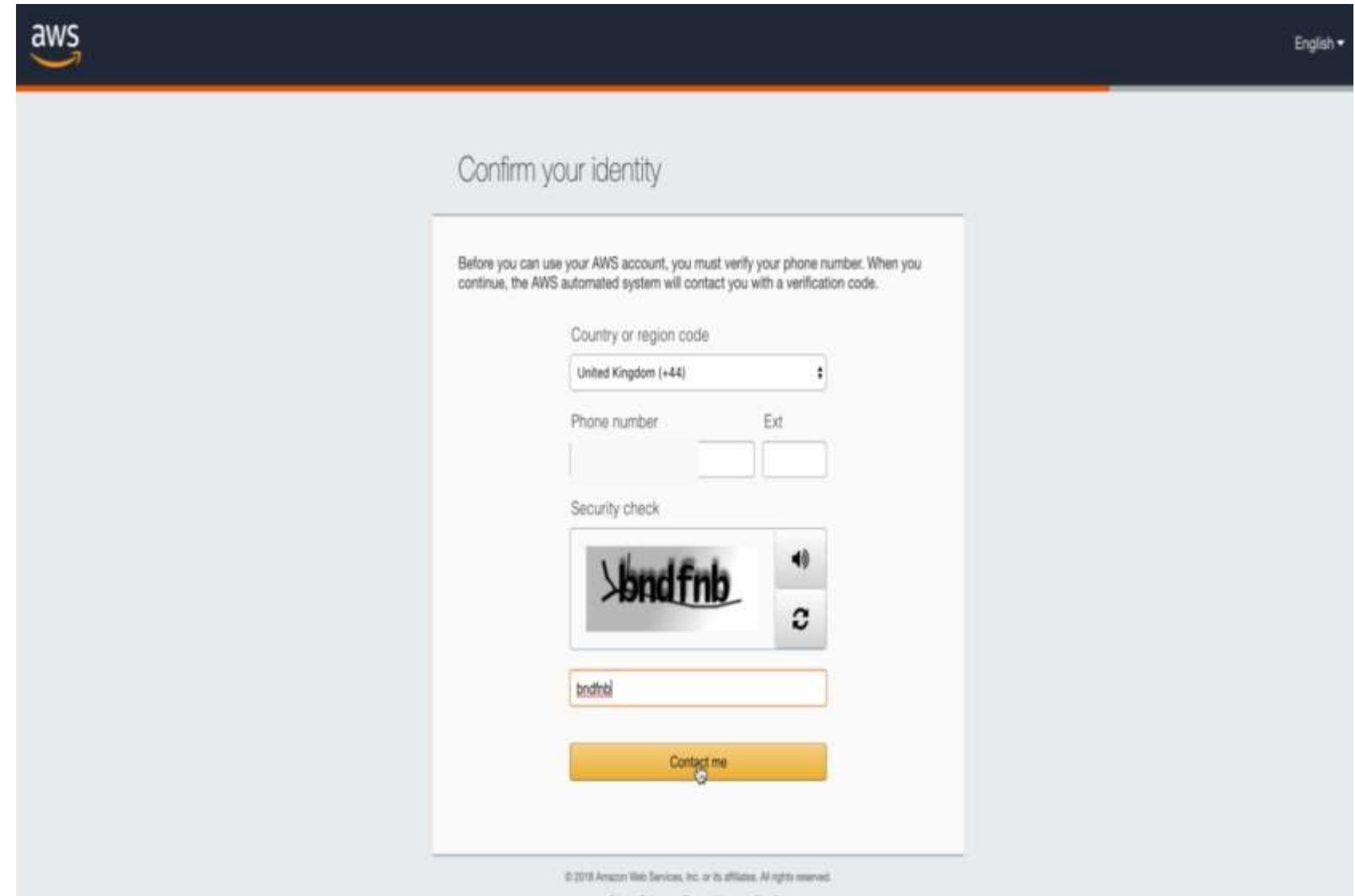
Expiration date

01 2019

Cardholder's name

How to SignUp to the AWS platform

- After providing your payment information, confirm your identity by entering your phone number and security check code, and then click on the "Contact me" button.
-



How to SignUp to the AWS platform

- AWS will contact you to verify whether the provided contact number is correct or not.

Call in progress...

Please answer the call from AWS and, when prompted, enter the 4-digit number on your phone keypad.

3 0 8 7



How to SignUp to the AWS platform

- When number is verified, then the following message appears on the screen.
- The final step is the confirmation step. Click on the link to log in again; it redirects you to the "**Management Console**".



What is IAM?

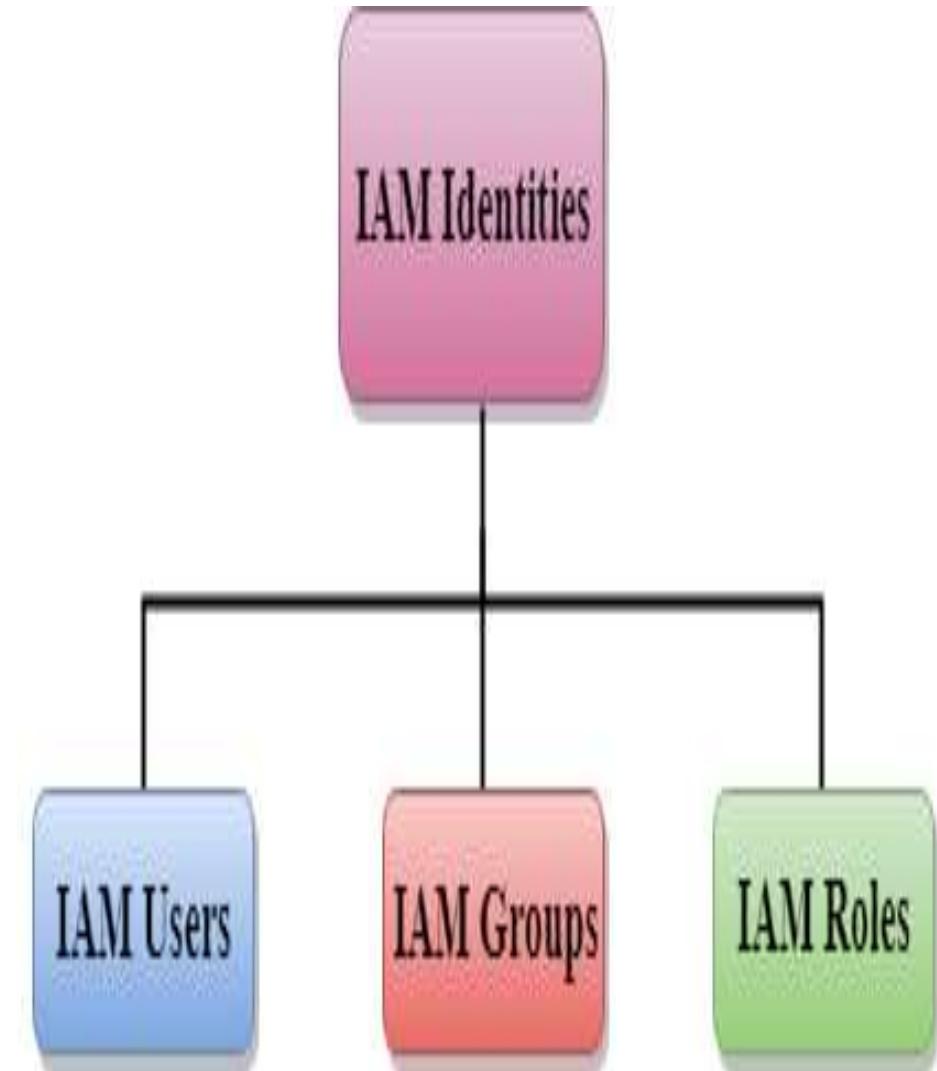
- IAM stands for **Identity Access Management**.
- IAM allows you **to manage users and their level of access to the aws console**.
- It is **used to set users, permissions and roles**.
- It allows you to **grant access to the different parts of the aws platform**.
- AWS Identity and Access Management is a web service that enables Amazon Web Services (AWS) customers **to manage users and user permissions in AWS**.
- With IAM, Organizations can centrally manage users, security credentials such as **access keys, and permissions that control which AWS resources users can access**.
- Without IAM, Organizations with multiple users must either create multiple user accounts, each with its own billing and subscriptions to AWS products or share an account with a single security credential.

What is IAM?

- Without IAM, you also **don't have control about the tasks that the users can do.**
- IAM enables the organization to **create multiple users, each with its own security credentials, controlled and billed to a single aws account.**
- IAM allows the user to do only what they need to do as a part of the user's job.

IAM Identities

- IAM identities are created to provide authentication for people and processes in your aws account.
- IAM Users
- IAM Groups
- IAM Roles



- **AWS Account Root User**
- When you first create an AWS account, you create an account as a root user identity which is used to sign in to AWS.
- You can sign to the AWS Management Console by entering your email address and password.
- The combination of email address and password is known as **root user credentials**.
- When you sign in to AWS account as a root user, you have unrestricted access to all the resources in AWS account.
- The Root user can also access the billing information as well as can change the password also.

- **IAM Users**
- An IAM User is an entity created in AWS that provides a way to interact with AWS resources.
- Purpose of IAM Users is that they can sign in to the AWS Management Console and can make requests to the AWS services.
- Newly created **IAM users** have no password and no access key.
- Newly created IAM Users do not have permissions, i.e., they are not authorized to access the AWS resources.
- Advantage of using individual IAM Users is that you can assign the permissions individually.
- We can even assign the administrative permissions, who can administer your AWS resources and also administer other IAM Users.
- Each IAM User is associated with one and only one AWS account

- **IAM Group**
- An IAM Group is a collection of users.
- Group specifies the permission for a collection of users, and it also makes it possible to manage the permissions easily for those users.
- **IAM Role**
- A role is a set of permissions that grant access to actions and resources in AWS. These permissions are attached to the role, not to an IAM User or a group.
- A role is not uniquely associated with a single person; it can be used by anyone who needs it.
- A role does not have long term security credential, i.e., password or security key.

- **What is the AWS Command Line Interface?**
- The AWS Command Line Interface (AWS CLI) is an **open source tool** that enables you to interact with AWS services using commands in your command-line shell.
- The AWS CLI enables you to start running commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from the command prompt in your terminal.

- AWS offers a wide range of storage services that can be provisioned depending on your project requirements and use case.
- AWS storage services have different provisions for highly confidential data, frequently accessed data, and the not so frequently accessed data.
- You can choose from various storage types namely, ***object storage, file storage, block storage services, backups, and data migration options.***

What is S3? [3]

- S3 stands for **Simple Storage Service**.
- S3 provides developers and IT teams with **secure, durable, highly scalable object storage**.
- It is easy to use with a simple web services interface to store and retrieve any amount of data from anywhere on the web.
- It is **Object-based storage**, i.e., you can **store the images, word files, pdf files**, etc.
- The files which are stored in S3 can be from **0 Bytes to 5 TB**.
- It has unlimited storage means that you can store the data as much you want.
- Files are stored in **Bucket**.
- **A bucket is like a folder available in S3 that stores the files**.
- S3 is a universal namespace, i.e., the names must be unique globally.
- Bucket contains a **DNS address**.
- The bucket must contain a **unique name to generate a unique DNS address**.
- There is a limit of **100 buckets per AWS accounts**. But it can be increased if requested from AWS support.

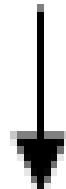
What is S3? [3]

- If you create a bucket, URL look like:

<https://s3-eu-west-1.amazonaws.com/acloudguru>



Region name



Bucket name

AWS S3 Terminology [3]

- **Bucket Owner:** The person or organization that owns a particular bucket is its *bucket owner*.
- **Import/Export Station:** A machine that uploads or downloads data to/from S3.
- **Key:** Key, in S3, is a unique identifier for an object in a bucket. For example in a bucket ‘ABC’ your *GFG.java* file is stored at *javaPrograms/GFG.java* then ‘*javaPrograms/GFG.java*’ is your object key for *GFG.java*.
- It is important to note that ‘bucketName+key’ is unique for all objects.
- Only one object for a key in a bucket.
- If you upload 2 files with the same key.
- The file uploaded latest will overwrite the previously contained file.

AWS S3 Terminology [3]

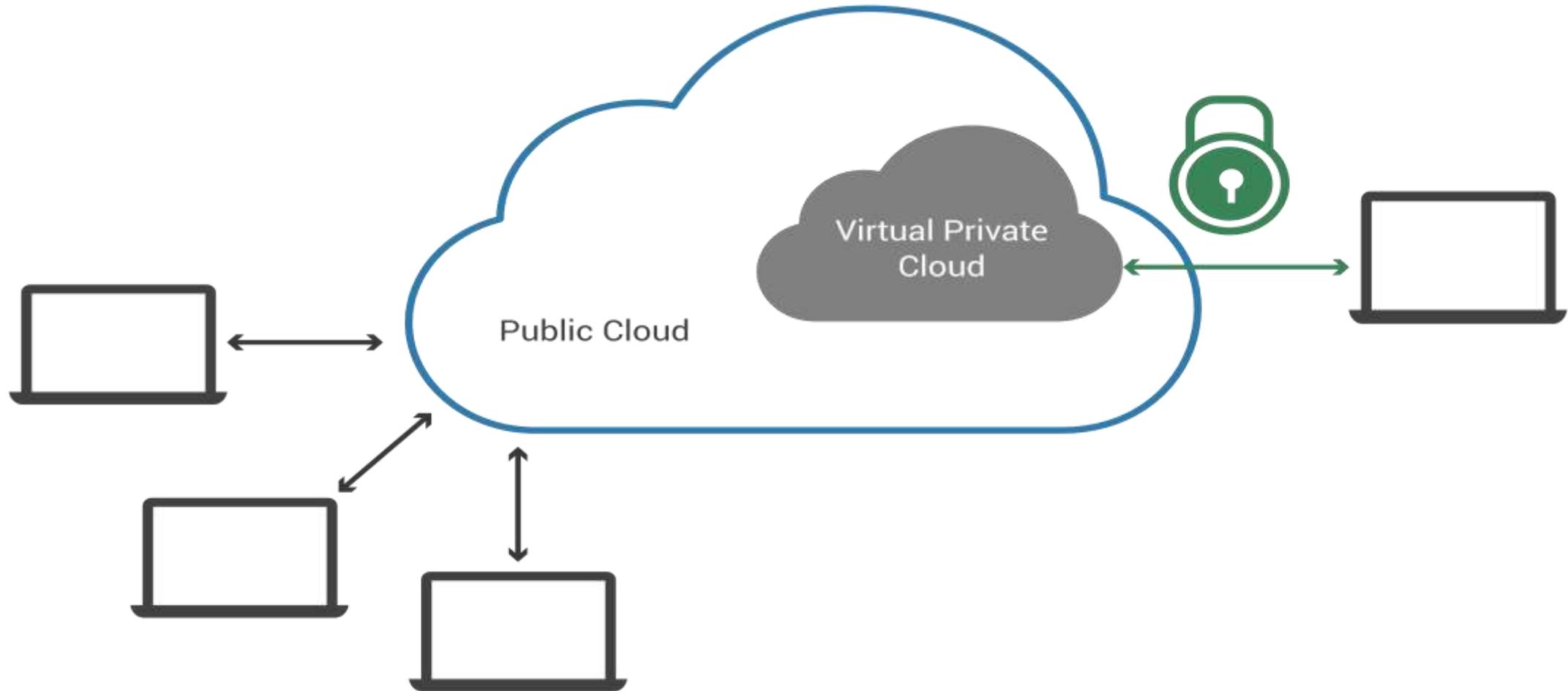
- **Versioning:**
- Versioning means to always **keep a record of previously uploaded files in S3**.
- Versioning is not enabled by default. Once enabled, it is enabled for all objects in a bucket.
- Versioning keeps all the copies of your file, so, it adds cost for storing multiple copies of your data.
- For example, 10 copies of a file of size 1GB will have you charged for using 10GBs for S3 space.
- Versioning is helpful to prevent unintended overwrites and deletions.
- Note that objects with the same key can be stored in a bucket if versioning is enabled (since they have a unique version ID).

AWS S3 Terminology

- **null Object:**
- Version ID for objects in a bucket where versioning is suspended is null.
- Such objects may be referred to as null objects.
- For buckets with versioning enabled, each version of a file has a specific version ID.
- **Object:** Fundamental entity type stored in AWS S3.
- **Access Control Lists (ACL):**
- A document for verifying the access to S3 buckets from outside your AWS account.
- Each bucket has its own ACL.
- **Bucket Policies:**
- A document for verifying the access to S3 buckets from within your AWS account, this controls which services and users have what kind of access to your S3 bucket.
- Each bucket has its own Bucket Policies.

- **Lifecycle Rules:**
- This is a cost-saving practice that can move your files to AWS Glacier (The AWS Data Archive Service) or to some other S3 storage class for cheaper storage of old data or completely delete the data after the specified time.
- **Features of AWS S3:**
- **Durability**
- **Availability**
- **Server-Side-Encryption (SSE):** AWS S3 supports three types of SSE models:
 - **SSE-S3:** AWS S3 manages encryption keys.
 - **SSE-C:** The customer manages encryption keys.
 - **SSE-KMS:** The AWS Key Management Service (KMS) manages the encryption keys.
 - **AWS-S3** is region-specific.

What is a virtual private cloud (VPC)?[6]



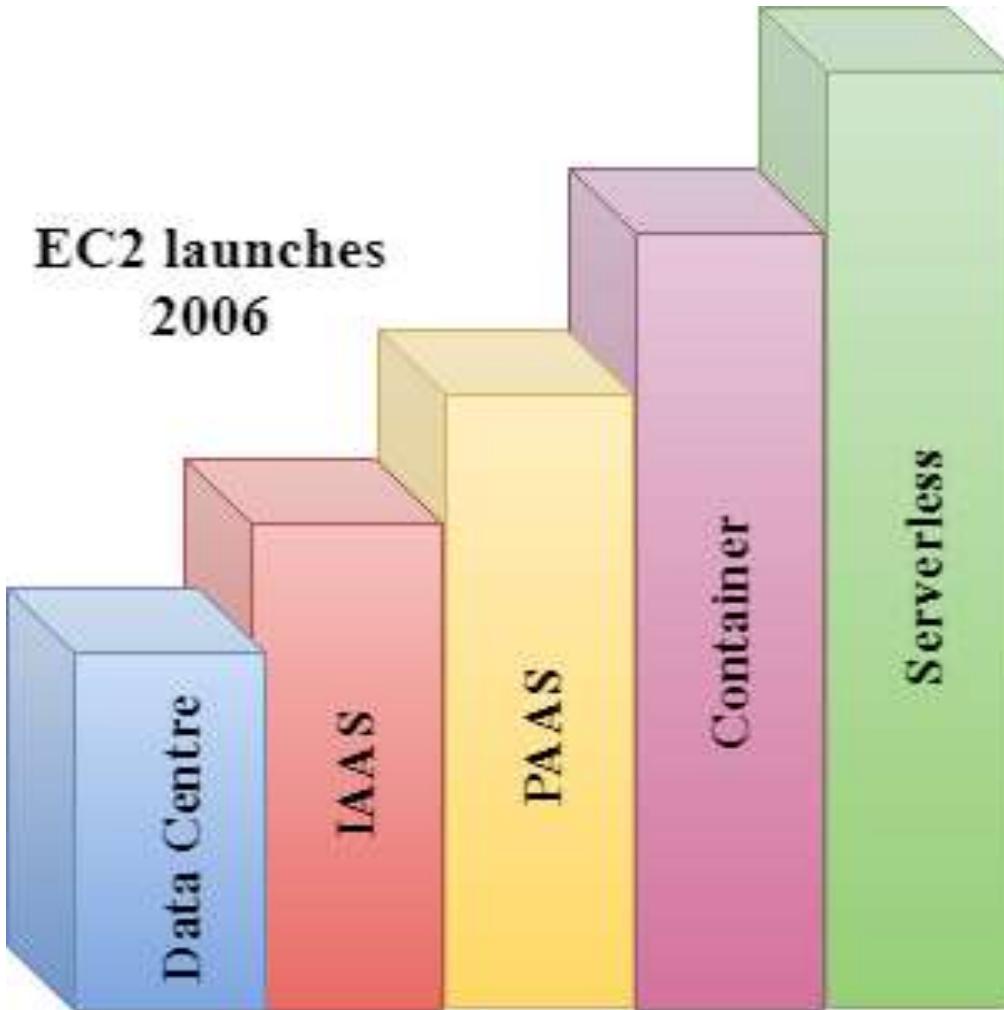
What is a virtual private cloud (VPC)? [6]

- A virtual private cloud (VPC) is a secure, isolated private cloud hosted within a public cloud.
- VPC customers can run code, store data, host websites, and do anything else they could do in an ordinary private cloud, but the private cloud is hosted remotely by a public cloud provider.
- VPCs combine the scalability and convenience of public cloud computing with the data isolation of private cloud computing.

What is a virtual private cloud (VPC)? [6]

- **What is a public cloud? What is a private cloud?**
- A public cloud is shared cloud infrastructure.
- Multiple customers of the cloud vendor access that same infrastructure, although their data is not shared – just like every person in a restaurant orders from the same kitchen, but they get different dishes.
- Public cloud service providers include AWS, Google Cloud Platform, and Microsoft Azure, among others.
- The technical term for multiple separate customers accessing the same cloud infrastructure is "multitenancy" (see [What Is Multitenancy?](#) to learn more).
- A private cloud, however, is single-tenant.
- A private cloud is a cloud service that is exclusively offered to one organization.
- A virtual private cloud (VPC) is a private cloud within a public cloud; no one else shares the VPC with the VPC customer.

What is Lambda?



- **Lambda:**
- Amazon released Lambda in 2015.
- You **do not have to take care of managing Data centre, managing infrastructure as a service, managing platform as a service or container.**
- You need to upload the code and Amazon will do everything for you.

What is Lambda?

- Lambda is **used to encapsulate Data centres, Hardware, Assembly code/Protocols, high-level languages, operating systems, AWS APIs.**
- Lambda is a **compute service where you can upload your code and create the Lambda function.**
- Lambda takes care of **provisioning and managing the servers used to run the code.**
- While using Lambda, you don't have to worry about **scaling, patching, operating systems, etc.**

How does AWS Lambda work? [4]



AWS Lambda Block Diagram

How does AWS Lambda work? [4]

- **Step 1:** First upload your AWS Lambda code in any language supported by AWS Lambda. [Java](#), Python, Go, and C# are some of the languages that are supported by AWS Lambda function.
- **Step 2:** These are some AWS services which allow you to trigger AWS Lambda.
- **Step 3:** AWS Lambda helps you to upload code and the event details on which it should be triggered.
- **Step 4:** Executes AWS Lambda Code when it is triggered by AWS services:
- **Step 5:** AWS charges only when the AWS lambda code executes, and not otherwise.

How does AWS Lambda work? [4]

- This will happen in the following scenarios:
 - Upload files in an S3 bucket
 - When HTTP get/post endpoint URL is hit
 - For adding/modifying and deleting Dynamo DB tables
 - In the process of data streams collection
 - Push notification
 - Hosting of website
 - Email sending

How does AWS Lambda work? [4]

- **Events that Trigger AWS Lambda**
- Here, are Events which will be triggered when you use AWS Lambda.
- Insert, updating and deleting data Dynamo DB table
- To include push notifications in SNS
- To search for log history in CloudTrail
- Entry into an S3 object
- DynamoDB can trigger AWS Lambda whenever there is data added, modified, and deleted in the table.
- Helps you to schedule the event to carry out the task at regular time pattern.
- Modifications to objects in S3 buckets
- Notifications sent from Amazon SNS.
- AWS Lambda can be used to process the CloudTrail logs
- API Gateway allows you to trigger AWS Lambda on GET/POST methods.

What is Elastic Compute Cloud (EC2)?

- EC2 stands for Elastic Compute Cloud.
- EC2 is on-demand computing service on the AWS cloud platform. U
- It also allows the user to configure their instances as per their requirements i.e. allocate the RAM, ROM, and storage according to the need of the current task
- EC2 has resizable capacity current task.
- EC2 offers security, reliability, high-performance and cost-effective infrastructure so as to meet the demanding business needs.

Features of Amazon EC2:

- **Functionality** : Amazon EC2 itself comes with a set of default AMI(Amazon Machine Image) options supporting various operating systems along with some pre-configured resources like RAM, ROM, storage, etc.
- **Operating Systems** – Amazon EC2 includes a wide range of operating systems to choose from while selecting your AMI.
- Currently, AWS has the following most preferred set of operating systems available on the EC2 console.
- **Amazon Linux, Windows Server, Ubuntu Server, SUSE Linux, Red Hat Linux**

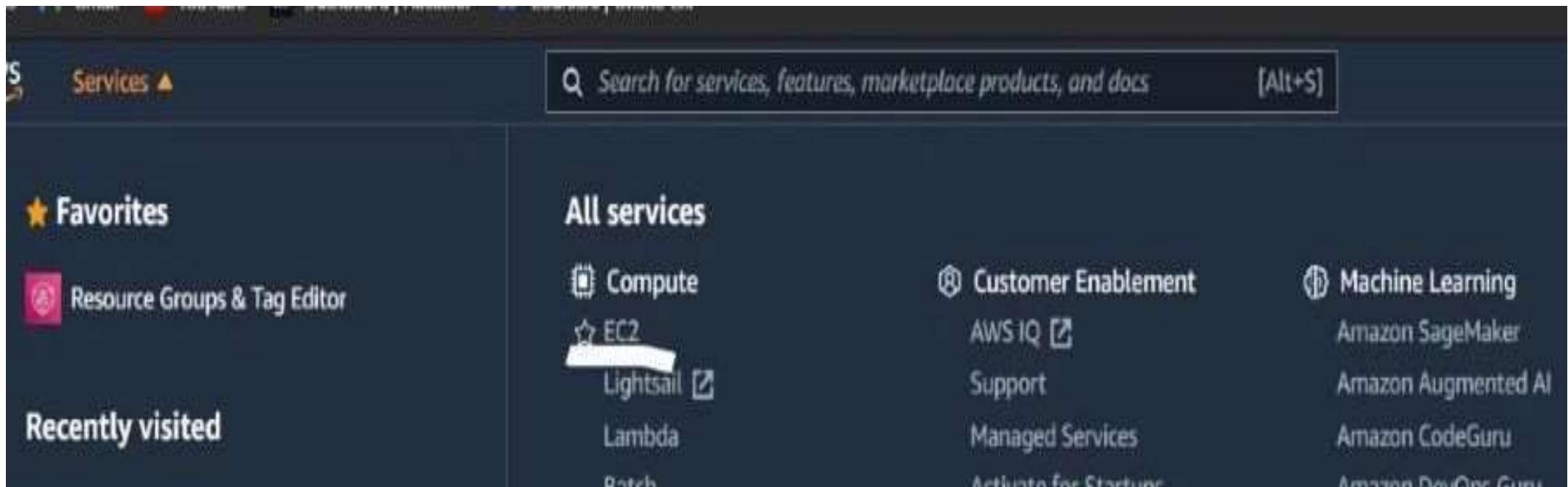


Features of Amazon EC2:

- **Software :**
- It allows its users to choose from various software present to run on their EC2 machines.
- Numerous software like SAP, LAMP and Drupal, etc are available on AWS to use.
- **Scalability and Reliability** – EC2 provides us the facility to scale up or scale down as per the needs.

Features of Amazon EC2:

- First login into your AWS account.
- Once you are directed to the management console. From the left click on “Services” and from the listed options click on EC2.



Features of Amazon EC2:

- Afterward, you will be redirected to the EC2 console. Here is the image attached to refer to various features in EC2.

The screenshot shows the AWS EC2 Dashboard. On the left, a sidebar menu includes 'New EC2 Experience' (with a 'Tell us what you think' link), 'EC2 Dashboard' (marked as 'New'), 'Events', 'Tags', 'Limits', 'Instances' (with sub-options like 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances' (marked as 'New'), 'Dedicated Hosts', 'Capacity Reservations'), and 'Images' (with 'AMIs'). The main content area has a blue header 'Welcome to the new EC2 console!' with a message about the redesign. Below it, the 'Resources' section displays a grid of metrics: Instances (running) 0, Dedicated Hosts 0, Elastic IPs 0, Instances 1, Key pairs 1, Load balancers 0, Placement groups 0, Security groups 3, Snapshots 0, and Volumes 0. To the right, the 'Account attributes' section lists 'Supported platforms' (VPC), 'Default VPC' (vpc-7d4a8f16), 'Settings', 'EBS encryption', 'Zones', 'Default credit specification', and 'Console experiments'. At the bottom, there's a callout for 'Microsoft SQL Server Always On availability groups'.

How to Launch a WordPress Website using Amazon EC2 Server ?

- Setting up WordPress website using an EC2 instance:
 1. Launch an EC2 instance:
- Go to your [AWS console](#) and sign in with your credentials.
- After login to the AWS account, select the EC2 service from the list of all services.

The screenshot shows the AWS Management Console homepage. On the left, there's a sidebar titled "AWS services" with sections for "Find Services" (a search bar), "Recently visited services" (empty), and "All services". Under "Compute", "EC2" is highlighted with a green border. Other services listed under Compute include Lightsail, ECR, ECS, EKS, Lambda, Batch, Elastic Beanstalk, Serverless Application Repository, AWS Outposts, and EC2 Image Builder. Under "Storage", S3, EFS, FSx, S3 Glacier, and Storage Gateway are listed. The main content area displays several service cards: "Blockchain" (Amazon Managed Blockchain), "Satellite" (Ground Station), "Quantum Technologies" (Amazon Braket), "Management & Governance" (AWS Organizations, CloudWatch, AWS Auto Scaling, CloudFormation, CloudTrail, Config, OpsWorks, Service Catalog), "Security, Identity, & Compliance" (IAM, Resource Access Manager, Cognito, Secrets Manager, GuardDuty, Inspector, Amazon Macie, AWS Single Sign-On, Certificate Manager, Key Management Service, CloudHSM, Directory Service, WAF & Shield, Artifact, Security Hub, Detective), "Amazon Redshift" (Fast, simple, cost-effective data warehouse that can extend queries to your data lake), "Run Serverless Containers with AWS Fargate" (AWS Fargate runs and scales your containers without having to manage servers or clusters), "Scalable, Durable, Secure Backup & Restore with Amazon S3" (Discover how customers are building backup & restore solutions on AWS that save money), and "AWS Marketplace" (Discover software and services available in the AWS Marketplace). A sidebar on the right titled "Access resources on the go" shows a mobile device icon and text about the AWS Console Mobile App.

How to Launch a WordPress Website using Amazon EC2 Server ?

- Click the **Launch instance** button to create an instance.

The screenshot shows the AWS EC2 Dashboard in the US East (Ohio) Region. The left sidebar includes links for New EC2 Experience, EC2 Dashboard, Events, Tags, Reports, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs), and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main content area displays EC2 Resources: Running instances (0), Dedicated Hosts (0), Volumes (1), Key pairs (3), Placement groups (0), Elastic IPs (0), Snapshots (0), Load balancers (0), and Security groups (4). A callout box provides information about launching Microsoft SQL Server Always On availability groups. The 'Launch instance' button is highlighted with a green border. The 'Service health' section indicates the service is operating normally in the US East (Ohio) Region.

Resources

You are using the following Amazon EC2 resources in the US East (Ohio) Region:

Running instances	0	Elastic IPs	0
Dedicated Hosts	0	Snapshots	0
Volumes	1	Load balancers	0
Key pairs	3	Security groups	4
Placement groups	0		

(i) Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. [Learn more](#) X

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance ▼

Note: Your instances will launch in the US East (Ohio) Region

Service health

Region: US East (Ohio) Status: This service is operating normally

How to Launch a WordPress Website using Amazon EC2 Server ?

- 2. Configuring EC2 instance:
- Search for WordPress in the marketplace tab (left side) in the dashboard and select the instance named as WordPress Certified by Bitnami and Automattic.

The screenshot shows the AWS Marketplace search results for "wordpress". The search bar at the top contains "wordpress". Below it, a navigation bar has tabs: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. The "1. Choose AMI" tab is selected. The main content area displays search results. A green box highlights the first result, "WordPress Certified by Bitnami and Automattic". This item is marked as "Free tier eligible". It has a 4.5-star rating from 114 reviews. The description states: "WordPress is the world's most popular content management platform. It includes the new Gutenberg editor and over 45,000 themes and plugins. This image is certified by Bitnami as secure, up-to-date, and packaged using industry best practices, and approved by Automattic, the experts behind WordPress." A "Select" button is visible. Below this, two more results are shown: "WordPress with NGINX and SSL Certified by Bitnami and Automattic" and "WordPress Multisite Certified by Bitnami and Automattic", each with its own "Select" button. On the left sidebar, there are filters for "Categories" (All Categories, Infrastructure-Software, DevOps, Business Applications, Industries), "Operating System" (All Linux/Unix), and "AWS Marketplace" (128 products). The "AWS Marketplace" filter is currently selected and highlighted with a green box.

How to Launch a WordPress Website using Amazon EC2 Server ?

- Now you'll see the pricing details in which you have to click continue.

Infrastructure as Code – What is it and Why is it important?

- Managing IT infrastructure was a manual process.
- People would physically put servers in place and configure them.
- Applications would be deployed only after the machines were configured to the correct settings required by the OS and applications.
- Unsurprisingly, this manual process would often result in several problems such as follows –
 - cost
 - scalability
 - availability
 - inconsistency

Infrastructure as Code – What is it and Why is it important?[7,8]

- Infrastructure as code (IaC) tools allow you to **manage infrastructure with configuration files rather than through a graphical user interface**.
- IaC allows you to **build, change, and manage your infrastructure** in a safe, consistent, and repeatable way by **defining resource configurations that you can version, reuse, and share[8]**.
- Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.
- The IT infrastructure managed by this comprises both physical equipment's such as bare-metal servers as well as virtual machines and associated configuration resources[7].

How IaC Works?

- The imperative approach “gives orders”.
- It defines a sequence of commands or instructions so the infrastructure can reach the final result.
- A declarative approach, on the other hand, “declares” the desired outcome.
- Instead of explicitly outlining the sequence of steps the infrastructure needs to reach the final result, the declarative approach shows what the final result looks like.

Types of IaC

Imperative

Declarative

Benefits of infrastructure as code

- **Speed**
- With Infrastructure as code, you can quickly set up your complete infrastructure by running a script.
- **Consistency**
- Manual processes sometimes result in mistakes. Manual infrastructure management will result in discrepancies, no matter how hard you try. IaC solves that problem.
- **Accountability**
- This one is quick and easy. Since you can version IaC configuration files like any source code file, you have full traceability of the changes each configuration suffered.



Benefits of infrastructure as code

- **Increased Efficiency**
- IaC can make the entire software development lifecycle more efficient. By employing infrastructure as code, you can deploy your infrastructure architectures in many stages.
- **Lower Cost**
- Lowering the costs of infrastructure management. By employing cloud computing along with IaC, you dramatically reduce your costs.
- Won't have to spend money on hardware, hire people to operate it, and build or rent physical space to store it.



Infrastructure-as-code tools AWS CloudFormation[7]



- CloudFormation permits users to model their infrastructure within a JSON or YAML template file.
- The service also adds automation features to help you with the deployment of resources in a repeatable and manageable way, and you only pay for the resources that you use.
- With the template configured to your application specifications, CloudFormation will take care of the rest of the tasks for you.
- The use of plaintext is very handy.
- Both YAML or JSON are supported, and it is easy to establish a secure infrastructure model at any complexity level from the many templates available from CloudFormation.

Azure Resource Manager[7]



- Using this tool, users are able to provision infrastructure and handle dependencies in one seamless cycle through Azure Resource Manager templates (ARM templates).

Google Cloud Deployment Manager [7]



- This tool bases its execution on config files such as YAML and templates (JINJA2 or PYTHON) all within the Google Cloud Platform.



- Terraform automation has various shapes and is orchestrated in varying degrees with the focus on the core plan/apply cycle.
- Some teams run Terraform locally but they use wrapper scripts to set up a consistent working directory for Terraform to run in.
- Other development teams may also run Terraform entirely within an alternate orchestration tool such as Jenkins.
- It is by far the most adaptable tool on this list but subsequently potentially intimidating, to begin with at least.

**CHEF**™

- Chef is a quite popular IaC tool among CI/CD practitioners.
- It uses Ruby-based DSL and this is certainly a huge plus.
- It has “cookbook” versioning from the beginning and allows you to maintain a consistent configuration.
- This is possible even when the infrastructure needs to keep up with the rapid growth of the app it hosts.
- Chef gives recipes and cookbooks at the heart of its configuration.
- These are self-styled appellations for templates and collections of templates that you can use out of the box.



- Ansible is a tool designed with the perspective of automation from the start.
- This tool focuses on providing “radically simple” configuration language as well as being able to manage cloud instances immediately with no modifications.
- It is also useful for performing arbitrary IT orchestration such as zero downtime rolling updates, hotfixes, and so on as opposed to being configuration management specific.
- Rather than managing systems as individual units, you just describe how components and the system in general interact with each other and Ansible will handle the rest.



- This tool runs the data centers for several significant companies like Reddit, Dell, and Google and runs on all OS systems.

Load Balancing in Cloud Computing

- Load balancing is the method that allows you to have a proper balance of the amount of work being done on different pieces of device or hardware equipment.
- The load of the devices is balanced between different servers or between the CPU and hard drives in a single cloud server.
- Benefit : To improve the speed and performance of each single device, and the other is to protect individual devices from hitting their limits by reducing their performance.
- Load balancing is beneficial with almost any type of service, such as HTTP, SMTP, DNS, FTP, and POP/IMAP.
- It also increases reliability through redundancy. A dedicated hardware device or program provides the balancing service.

Load Balancing in Cloud Computing

- Different Types of Load Balancing Algorithms in Cloud Computing:

1. Static Algorithm

- Static algorithms are built for systems with very little variation in load.
- The entire traffic is divided equally between the servers in the static algorithm.
- This algorithm requires in-depth knowledge of server resources for better performance of the processor, which is determined at the beginning of the implementation.
- The decision of load shifting does not depend on the current state of the system.

2. Dynamic Algorithm

- First finds the lightest server in the entire network and gives it priority for load balancing.
- This requires real-time communication with the network which can help increase the system's traffic. Here, the current state of the system is used to control the load.

3. Round Robin Algorithm

- Uses round-robin method to assign jobs.
- First, it randomly selects the first node and assigns tasks to other nodes in a round-robin manner.
- This is one of the easiest methods of load balancing.
- Processors assign each process circularly without defining any priority.
- It gives fast response in case of uniform workload distribution among the processes.
- All processes have different loading times.

4. Weighted Round Robin Load Balancing Algorithm

- Developed to enhance the most challenging issues of Round Robin Algorithms.

Load Balancing in Cloud Computing

- In this algorithm, there are a specified set of weights and functions, which are distributed according to the weight values.
- Processors that have a higher capacity are given a higher value.
- Therefore, the highest loaded servers will get more tasks.
- When the full load level is reached, the servers will receive stable traffic.

5. Opportunistic Load Balancing Algorithm

- Allows each node to be busy.
- It never considers the current workload of each system.
- OLB distributes all unfinished tasks to these nodes.

Types of Load Balancing

- Network Load Balancing
- HTTP(S) load balancing
- Internal Load Balancing
- Hardware Load Balancer
- Software Load Balancer
- Virtual Load Balancer

WHY CLOUD LOAD BALANCING IS IMPORTANT IN CLOUD COMPUTING?

- Offers better performance
- Helps Maintain Website Traffic
- Can Handle Sudden Bursts in Traffic
- Greater Flexibility

- Systems Manager is an AWS service that lets customers centrally manage their EC2 instances to ensure security and compliance are in place based on the organizational policies.
- Customers can also make use of the service to manage on-premises VM's as well as the VM's hosted on hybrid cloud environments.
- Systems Manager makes use of SSM agent, a piece of software that can be installed and configured on virtual machines running Linux, Windows, MacOS, and Raspbian operating systems.
- **How Systems Manager works**

References

- [1] <https://www.javatpoint.com/advantages-and-disadvantages-of-cloud-computing>
- [2] <https://aws.amazon.com/what-is-cloud-computing/>
- [3] <https://www.geeksforgeeks.org/introduction-to-aws-simple-storage-service-aws-s3/>
- [4] <https://www.guru99.com/aws-lambda-function.html>
- [5] <https://blog.vsoftconsulting.com/blog/aws-lambda-function-how-it-works-and-how-to-create-it>
- [6] <https://www.cloudflare.com/learning/cloud/what-is-a-virtual-private-cloud/>
- [7] <https://www.edureka.co/blog/infrastructure-as-code/>
- [8] <https://developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code>
- [9] <https://www.javatpoint.com/load-balancing-in-cloud-computing>

Presentation Topic

Unit IV - Amazon Web Services

Vishal Ambadas Meshram

vishal.meshram@viit.ac.in

Department of Computer Engineering



BRACT'S, Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)





Cloud Computing:

The practice of using network of remote servers hosted on the Internet to store, manage, and process data rather than a local server or a personal computer.

Cloud Concept



On-Premise

1. You own the Server.
2. You hire IT people.
3. You pay or rent the real-estate.
4. You take all the risk.

Cloud



1. Someone else owns the Server.
2. Someone else hire IT people.
3. Someone else pay or rent the real-estate.
4. You are responsible for configuring your cloud services & code.
Someone else takes care of the rest.

Cloud Computing Models

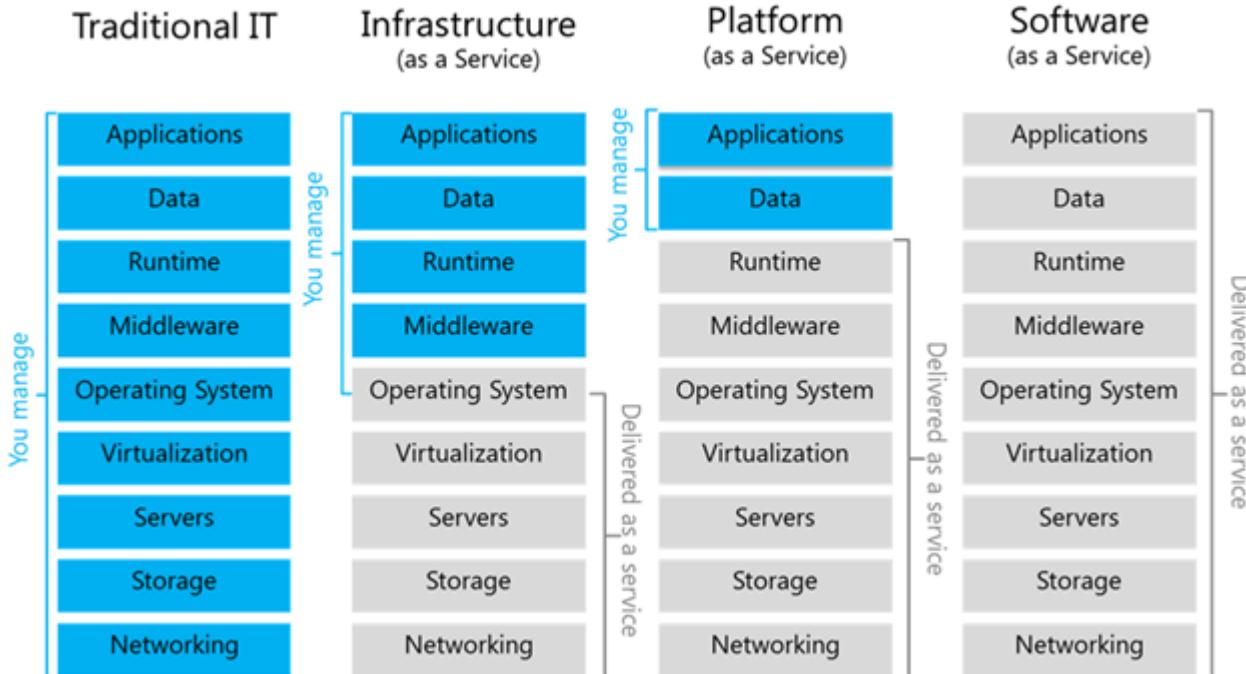


Fig Ref : <https://dachou.github.io/2018/09/28/cloud-service-models.html>

Cloud Computing Deployment Models

Cloud

Fully utilizing cloud computing

Hybrid

Using both Cloud and On-Premise

On-Premise

Deploying resources on-premises, using virtualization and resource management tools, is sometimes called “private cloud”.



- Startups
- SaaS offerings
- New projects and companies



- Banks
- FinTech, Investment Management
- Large Professional Service providers
- Legacy on-premise



- Public Sector eg. Government ACTIVATE WINDOWS
Go to Settings to activate Windows.
- Super Sensitive Data eg. Hospitals
- Large Enterprise with heavy regulation eg. Insurance Companies

Contents:

- **What is AWS**
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- Compute services
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)
- IAM

What is AWS?

aws

Link: https://www.youtube.com/watch?v=a9__D53WsUs

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- Compute services
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)
- IAM

AWS Global Infrastructure Map

AWS serves over a million active customers in more than **190 countries**. AWS now spans **77 Availability Zones** within **24 geographic regions** around the world and has announced plans for 18 more Availability Zones and 6 more AWS Regions in Australia, India, Indonesia, Japan, Spain, and Switzerland.



AWS Global Infrastructure Map

Regions

- Physical Location in the world with multiple Availability Zones (AZ's).
- It's a geographical distinct location.
- Every region is physically isolated from and independent of every other region in terms of location, power, water supply.
- Each region has at least 2 AZ's
- AWS largest region is "**US-EAST**" (**Northern Virginia**)
- US-EAST1 is the region where you see all your billing information.

Availability Zones

- Availability Zones consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities.
- AZ's offer you the ability to operate production applications and databases that are more highly available, fault tolerant, and scalable than would be possible from a single data center.
- AZ's are represented by region code, followed by a letter identifier. Ex. "us-east-1a".
- "MultiAZ", distributing your instance across multiple AZ's allows failover configuration for handling request when goes down.

Edge Locations

- **An Edge location is a data center owned by a trusted partner of AWS which has a direct connection to the AWS network.**
- These location serve requests for CloudFront & Route 53. Requests going to either of these services will be routed to the nearest edge location automatically.
- S3 transfer acceleration traffic & API Gateway endpoint traffic also use the AWS Edge network.

“Customers are increasingly choosing AWS to host their cloud-based infrastructure and realize increased performance, security, reliability, and scale wherever they go. For the tenth year in a row, AWS is evaluated as a Leader in the [2020 Gartner Magic Quadrant for Cloud Infrastructure and Platform Services](#), placed highest in both axes of measurement—Ability to Execute and Completeness of Vision—among the top 7 vendors named in the report.”



Figure 1. Magic Quadrant for Cloud Infrastructure and Platform Services



Gartner, Magic Quadrant for Cloud Infrastructure & Platform Services, Raj Bala, Bob Gill, Dennis Smith, David Wright, Kevin Ji, 1 September 2020. This graphic was published by Gartner, Inc. as part of a larger research document and should be evaluated in the context of the entire document. The Gartner document is available upon request from AWS. Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

Lower Costs with AWS Up-Front and Increase Savings as Your Usage Grows

1

Replace up-front capital expense with low variable cost

“Average of 400 servers replaced per customer”

2

Economies of scale have allowed us to consistently lower costs

44 Price Reductions

3

Pricing model choice to support variable & stable workloads

On-demand
Reserved
Spot

4

Save more money as you grow bigger

Tiered Pricing
Volume Discounts



Source: IDC Whitepaper, sponsored by Amazon, “The Business Value of Amazon Web Services Accelerates Over Time.” July 2012

4X More Reliable & 1/4 the Cost of On-Premises Infrastructure

END USERS BENEFITED FROM FEWER SERVICE DISRUPTIONS AND QUICKER RECOVERY ON AMAZON CLOUD INFRASTRUCTURE,
REDUCING DOWNTIME BY 72%

AMAZON CLOUD INFRASTRUCTURE REPRESENTS A
70% SAVINGS COMPARED WITH ON-PREMISE SOLUTIONS



WHITE PAPER

The Business Value of Amazon Web Services Accelerates Over Time

Sponsored by: Amazon

Randy Perry Stephen D. Hendrick
July 2012

EXECUTIVE SUMMARY

In early 2012, IDC interviewed 11 organizations that deployed applications on Amazon cloud infrastructure services. The purpose of the IDC analysis was to understand the economic impact of Amazon cloud infrastructure services over time, beyond the well-documented benefit of reduction in capex and opex. Specifically,

IDC set out to understand the long-term economic implications of moving workloads onto Amazon cloud infrastructure services, the impact of moving applications on developer productivity and business agility, and the new opportunities that businesses could address by moving resources onto Amazon cloud infrastructure services. The organizations interviewed ranged from small and medium-sized companies to companies with as many as 160,000 employees. Organizations in our study had been Amazon Web Services (AWS) customers for as few as seven months to as many as 5.3 years. Our interviews were designed to elicit both quantifiable information and anecdotes so that IDC could interpret the full return-on-investment (ROI) impact of Amazon cloud infrastructure services on these organizations. The study represents a broad range of

Business Value Highlights: Applications Running on AWS

- Five-year ROI: 625%
- Payback period: 7.1 months
- Software development productivity increase: 507%
- Average savings per application: \$518,990
- Downtime reduction: 72%
- IT productivity increase: 52%
- Five-year TCO savings: 70%



Architected for Enterprise Security Requirements

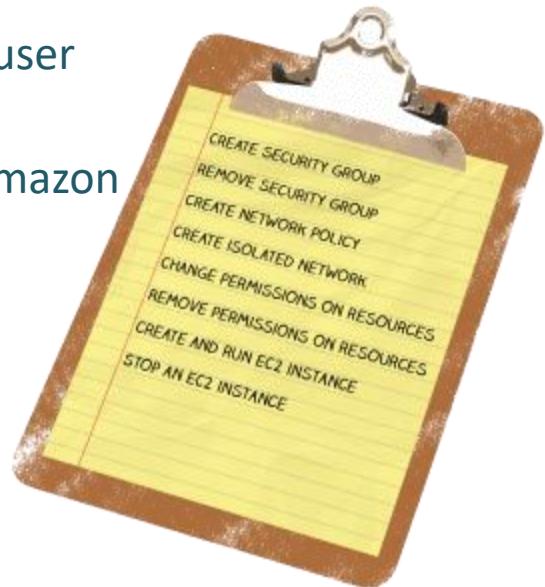
Certifications and accreditations for workloads that matter



AWS CloudTrail - AWS API call logging for governance & compliance

Log and review user activity

Stores data in Amazon S3, or archive to Amazon Glacier



Increased agility has become the
#1 reason businesses use the AWS
cloud



Enterprises Can't Afford to be Slow

Old World: Infrastructure in Weeks



AWS: Infrastructure in Minutes



- Add New Dev Environment
- Add New Prod Environment
- Add New Environment in Japan
- Add 1,000 Servers
- Remove 1,000 servers
- Deploy 2 PB Data warehouse
- Shut down 2 PB Data warehouse

Everything changes with this kind of agility

A Culture of Innovation: Experiment Often and Fail Without Risk



On-Premises

Experiment Infrequently

Failure is expensive

Less Innovation



Experiment Often

Fail quickly at a low cost

More Innovation

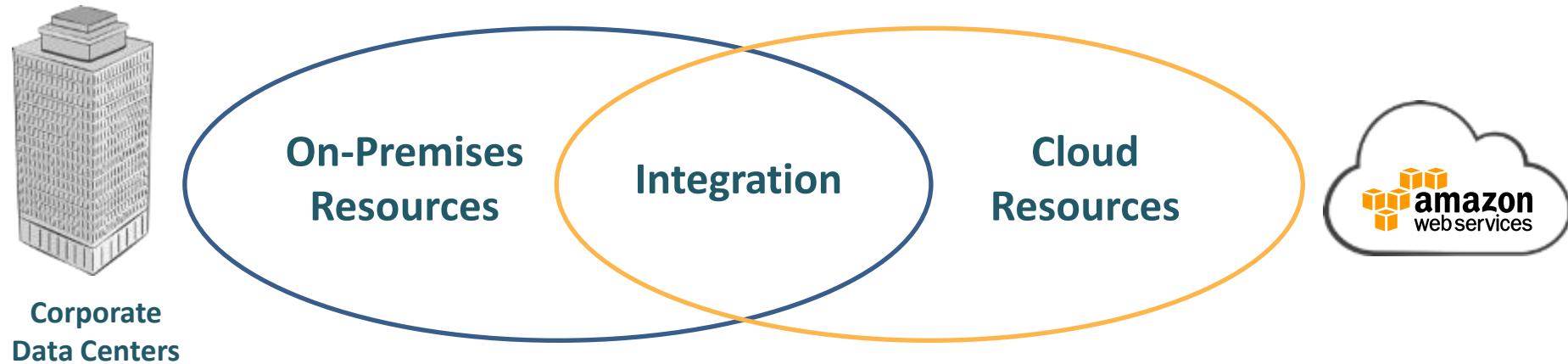
Many Enterprises Worry That These are the Only Two Choices

Build a
“private”
cloud

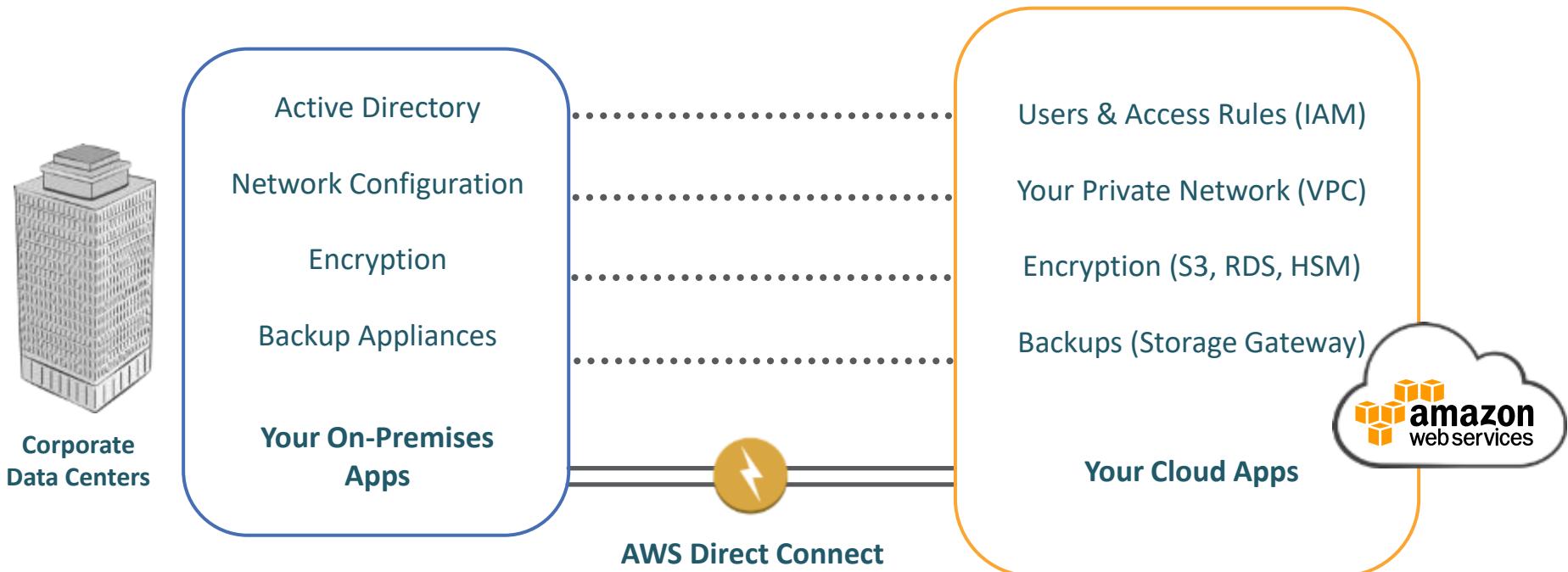


Rip everything out
and move to
AWS

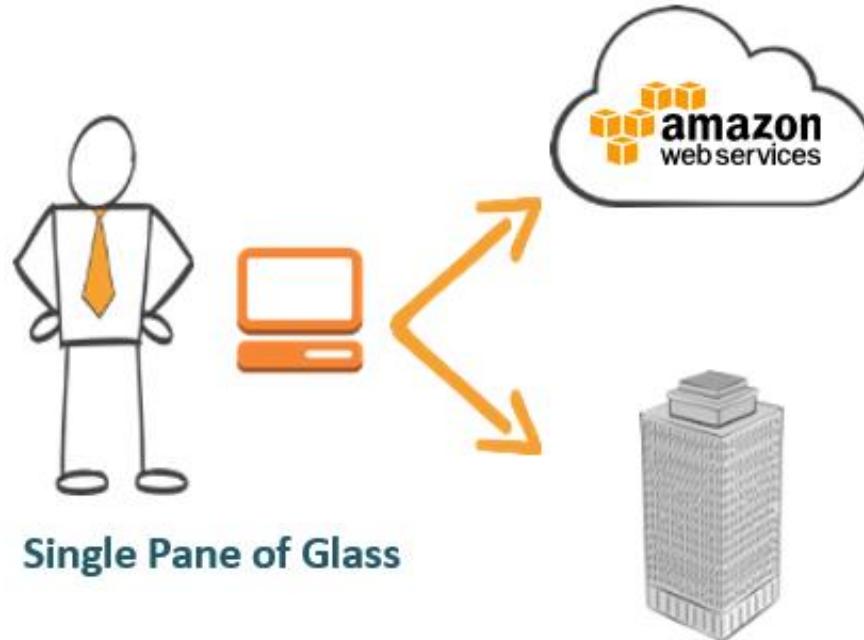
The Good News is that Cloud isn't an 'All or Nothing' Choice



Integrating AWS with Your Existing On-Premises Infrastructure



Tools to Help Customers Manage Resources Across Environments



Enterprises Use Cases on AWS

**Enterprise Apps
and Dev./Test**

**Big Data and
HPC**

**Storage, Backup
and Archival**

**Web, Mobile, and
Social Apps**

**Disaster
Recovery**

**Virtual
Desktops**

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- **AWS Account setup and billing alarms**
- Compute services
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)
- IAM

AWS billing alarms

1. Billing Preferences

2. Budget

3. Cloud Watch

Spend Summary Cost Explorer

Welcome to the AWS Billing & Cost Management console. Your last month, month-to-date, and month-end forecasted costs appear below.

Current month-to-date balance for February 2021, the exchange rate for the Payment Currency is estimated.

0.00 USD which converts to

0.00 INR

at today's exchange rate of 73.48255

Bills

\$1.6	\$1.59
\$1.2	
\$0.8	

Preferences

Billing preferences

Month-to-Date Spend by Service

The chart below shows the proportion of total monthly spend by service.

My Account 933126026785

My Organization

My Service Quotas

My Billing Dashboard

My Security Credentials

Sign Out

This screenshot shows the AWS Billing & Cost Management dashboard. On the left, a sidebar lists navigation options like Home, Cost Management, Cost Explorer, Budgets, and Billing preferences. The main area displays a Spend Summary with a current balance of 0.00 INR at an exchange rate of 73.48255. It also shows a bar chart of monthly spend by service, with one service costing \$1.59. The top right shows user information (vishal) and a Global dropdown menu. A red box highlights the 'Budgets' option in the sidebar and the 'My Billing Dashboard' link in the global menu.

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- **IAM Introduction**
- Compute services (EC2)
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)

IAM Introduction

- IAM stands for Identity Management Service.
- IAM is global service.
- Instead of using root account create a IAM user.
- Never share your root or IAM credentials.
- Policies are written in JSON (JavaScript Object Notation)

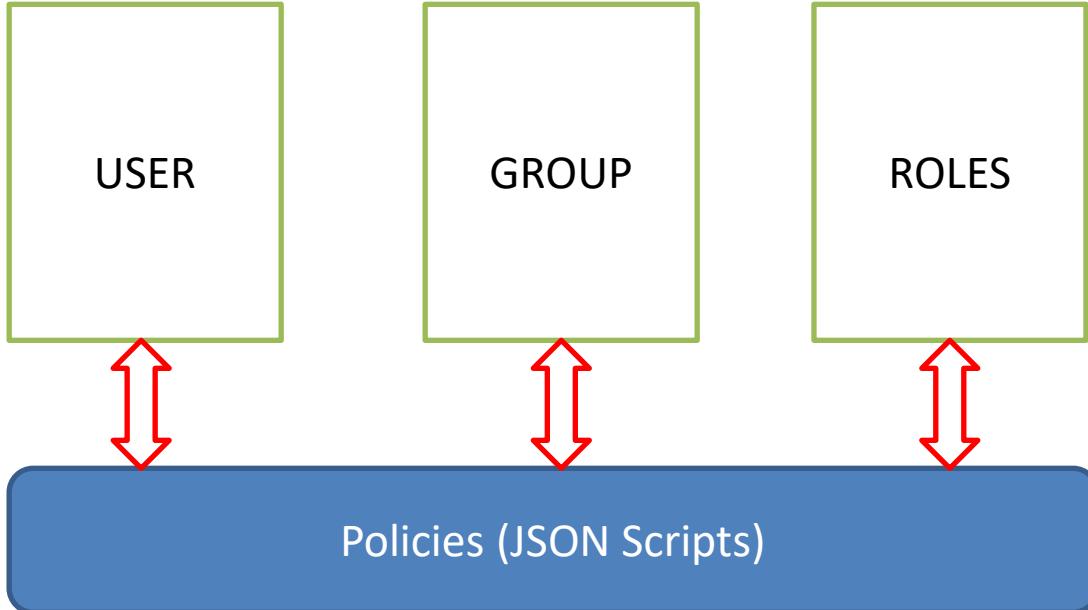
The screenshot shows the AWS IAM dashboard. On the left, a sidebar menu is open, with the 'Identity and Access Management (IAM)' section highlighted by a red box. The main content area displays the 'IAM dashboard' with the following sections:

- Sign-in URL for IAM users in this account:** <https://viit.signin.aws.amazon.com/console> (with Edit and Delete alias options)
- IAM resources:**
 - Users: 1
 - Groups: 1
 - Customer managed policies: 0
 - Roles: 6
 - Identity providers: 0
- Security alerts:** A warning message: "The root user for this account does not have Multi-factor authentication (MFA) enabled. Enable MFA to improve security for this account."
- Best practices:**
 - Grant least privilege access: Establishing a principle of least privilege ensures that identities are only permitted to perform the most minimal set of functions necessary to fulfill a specific task, while balancing usability and efficiency.
 - Use AWS Organizations: Centrally manage and govern your environment as you scale your AWS resources. Easily create new AWS accounts, group accounts to organize your workflows, and apply policies to accounts or groups for governance.
 - Enable Identity federation: Manage users and access across multiple services from your preferred identity source. Using AWS Single Sign-On centrally manage access to multiple AWS accounts and provide users with single sign-on access to all their assigned accounts from one place.
 - Enable MFA: For extra security, we recommend that you require multi-factor authentication (MFA) for all users.

On the right side, there are additional links and tools:

- Additional information: IAM documentation, Videos, IAM release history and additional resources.
- Tools: Web identity federation playground, Policy simulator.
- Quick links: My access key.
- Related services: AWS Organizations, AWS Single Sign-on (SSO).

AWS account ID:
933126026785



The screenshot shows the AWS IAM Security Credentials page. The left sidebar lists various IAM management options like Dashboard, Access management, and Policies. A search bar at the top has 'iam' typed into it. The main content area is titled 'Your Security Credentials' and provides instructions for managing AWS credentials. It highlights 'Password' and 'Multi-factor authentication (MFA)' with red boxes. A vertical navigation bar on the right includes links for My Account, My Organization, My Service Quotas, My Billing Dashboard, and My Security Credentials, with the latter also highlighted by a red box. The 'Global' button in the top right is also outlined in red.

aws Services ▾

Identity and Access Management (IAM)

Dashboard

Access management

Groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Search IAM

Q iam X

vishal Global Support

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, see [Managing IAM User Credentials](#).

To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#) in AWS General Reference.

▼ Password

You use an email address and password to sign in to secure pages on AWS, such as the AWS Management Console, AWS Forum, and AWS CLI. Your password should be a strong password that contains many characters, including numbers and punctuation. Store your password securely, do not share it, and do not reuse it.

[Click here](#) to change the password, name, or email address for your root AWS account.

▲ Multi-factor authentication (MFA)

▲ Access keys (access key ID and secret access key)

▲ CloudFront key pairs

▲ X.509 certificate

▲ Account identifiers

My Account 933126026785

My Organization

My Service Quotas

My Billing Dashboard

My Security Credentials

Sign Out

For MFA: Install “Google Authenticator” App.

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- IAM Introduction
- **Compute services (EC2)**
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)

EC2 (Elastic Compute Cloud)

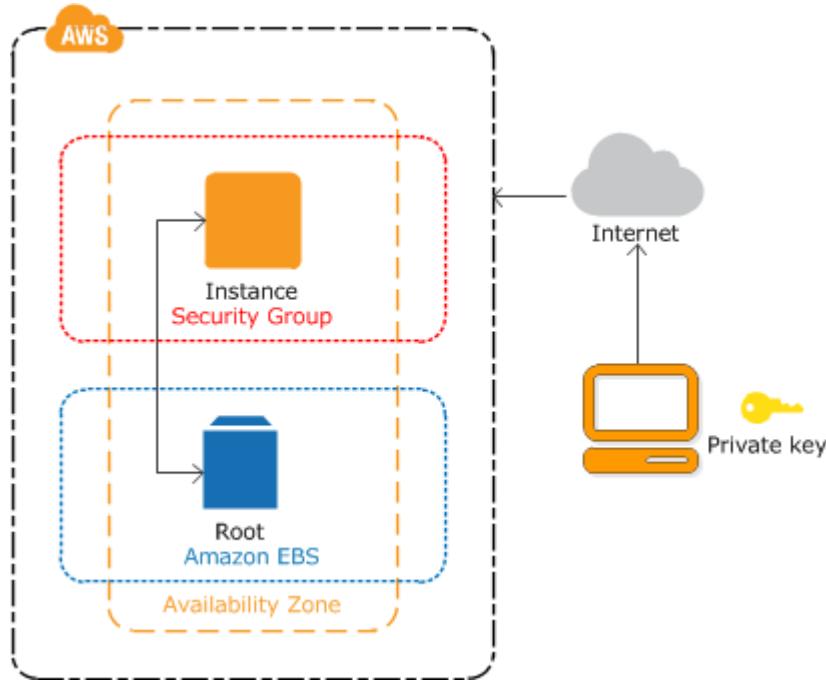
- ❖ EC2 stands for Elastic Compute Cloud.
- ❖ EC2 is a virtual machine on AWS.
- ❖ As per amazon “Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2’s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment.”



Get started with Amazon EC2 Linux instances



- Overview



Steps:

- Step 1: Launch an instance
- Step 2: Connect to your instance
- Step 3: Clean up your instance

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.



Step 1: To launch an instance

- 1) Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
- 2) From the console dashboard, choose **Launch Instance**.
- 3) The **Choose an Amazon Machine Image (AMI)** page displays a list of basic configurations, called *Amazon Machine Images (AMIs)*, that serve as templates for your instance. Select an HVM version of Amazon Linux 2. Notice that these AMIs are marked "Free tier eligible."
- 4) On the **Choose an Instance Type** page, you can select the hardware configuration of your instance. Select the t2.micro instance type, which is selected by default. The t2.micro instance type is eligible for the free tier. In Regions where t2.micro is unavailable, you can use a t3.micro instance under the free tier. For more information, see [AWS Free Tier](#).
- 5) Choose **Review and Launch** to let the wizard complete the other configuration settings for you.
- 6) On the **Review Instance Launch** page, under **Security Groups**, you'll see that the wizard created and selected a security group for you. You can use this security group, or alternatively you can select the security group that you created when getting set up using the following steps:
 - a) Choose **Edit security groups**.
 - b) On the **Configure Security Group** page, ensure that **Select an existing security group** is selected.
 - c) Select your security group from the list of existing security groups, and then choose **Review and Launch**.
- 7) On the **Review Instance Launch** page, choose **Launch**.
- 8) When prompted for a key pair, select **Choose an existing key pair**, then select the key pair that you created when getting set up.
- 9) A confirmation page lets you know that your instance is launching. Choose **View Instances** to close the confirmation page and return to the console.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

aws Services ▾

Search for services, features, marketplace products, and docs [Alt+S]

vishal Mumbai

New EC2 Experience Tell us what you think

EC2 Dashboard New

Events

Tags

Limits

Instances

- Instances New
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity Reservations

Images

- AMIs

Elastic Block Store

- Volumes

Resources

You are using the following Amazon EC2 resources in the Asia Pacific (Mumbai) Region:

Instances (running)	0	Dedicated Hosts	0
Elastic IPs	0	Instances	2
Key pairs	1	Load balancers	1
Placement groups	0	Security groups	3
Snapshots	1	Volumes	0

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance ▾

Service health

C Service Health Dashboard

Region Status

Asia Pacific (Mumbai) This service is operating

Supported platforms VPC

Default VPC vpc-30308859

Settings

EBS encryption

Zones

Default credit specification

Console experiments

Explore AWS

Enable Best Price-Performance with AWS Graviton2

AWS Graviton2 powered EC2 instances enable up to 40% better price performance for a broad spectrum of cloud workloads. Learn more

Get Up to 40% Better Price

The screenshot shows the AWS EC2 Dashboard. The left sidebar lists various EC2-related services like Instances, Images, and Elastic Block Store. The main content area displays a summary of resources in the Asia Pacific (Mumbai) Region, with a callout for launching an instance. The top right shows account attributes and a user profile. A red box highlights the 'Instances (running)' count of 0 and the 'Launch instance' button.

1. Search for “ec2” service.
2. Click on “Launch Instance”.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

[Cancel and Exit](#)

Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs

Free tier only ⓘ

AMI Name	Description	Root device type	Virtualization type	ENI Enabled	Action
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-08e0ca9924195beba (64-bit x86) / ami-0437d5dbe8fdc3d52 (64-bit Arm)	Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.	ebs	hvm	Yes	Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0a9d27a9f4f5c0efc (64-bit x86) / ami-0816d75a127c17a49 (64-bit Arm)	Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type	ebs	hvm	Yes	Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0b3acf3edf2397475 (64-bit x86) / ami-0ab71076ab9b53b0d (64-bit Arm)	SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available.	ebs	hvm	Yes	Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-073c8c0760395aab8 (64-bit x86) / ami-029dbbe5a11f53cf7 (64-bit Arm)	Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).	ebs	hvm	Yes	Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)

3. Select “Amazon Linux 2 AMI (HVM).
4. Click on Select.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, ~ 1 GiB memory, EBS only)

	Family	Type	vCPUs ⓘ	Memory (GiB) ⓘ	Instance Storage (GB) ⓘ	EBS-Optimized Available ⓘ	Network Performance ⓘ	IPv6 Support ⓘ
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel

Previous

Review and Launch

Next/Configure Instance Details

3. Select “t2.micro”
4. Click on “Configure Instance Details”.

Services ▾ Search for services, features, marketplace products, and docs [All 13]

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances Launch into Auto Scaling Group

Purchasing option Request Spot instances

Network vpc-30308859 (default) Create new VPC

Subnet No preference (default subnet in any Availability Zone) Create new subnet

Auto-assign Public IP No preference (default subnet in any Availability Zone)

subnet-af65abd4 | Default in ap-south-1c
subnet-db9613b2 | Default in ap-south-1a
subnet-883cc7c5 | Default in ap-south-1b

Placement group

Capacity Reservation Open

Domain join directory No directory Create new directory

IAM role None Create new IAM role

CPU options Specify CPU options

Shutdown behavior Stop

Stop - Hibernate behavior Enable hibernation as an additional stop behavior

Cancel Previous Review and Launch Next: Add Storage

5. Click on “Add Storage”.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type <i>i</i>	Device <i>i</i>	Snapshot <i>i</i>	Size (GiB) <i>i</i>	Volume Type <i>i</i>	IOPS <i>i</i>	Throughput (MB/s) <i>i</i>	Delete on Termination <i>i</i>	Encryption <i>i</i>
Root	/dev/xvda	snap-07e0efc01c68d3978	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

[Cancel](#) [Previous](#) [Review and Launch](#) **Next: Add Tags**

5. Click on “Add Tags”.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances ⓘ	Volumes ⓘ	Network Interfaces ⓘ
Name	My-First-Instance	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

Cancel

Previous

Review and Launch

Next: Configure Security Group

5. Click on “Configure Security Group”.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group

Select an existing security group

Security group name:

My-SG

Description: launch-wizard-1 created 2021-02-14T17:27:05.680+05:30

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	allow traffic from all IP addresses

Add Rule



Warning

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Previous **Review and Launch**
Go to Settings to activate Windows.

5. Click on “Review and launch”.



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.

⚠ Improve your instances' security. Your security group, My-SG, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.

You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

AMI Details

[Edit AMI](#)**Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-08e0ca9924195beba****Free tier eligible**

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is a...

Root Device Type: ebs Virtualization type: hvm

Instance Type

[Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

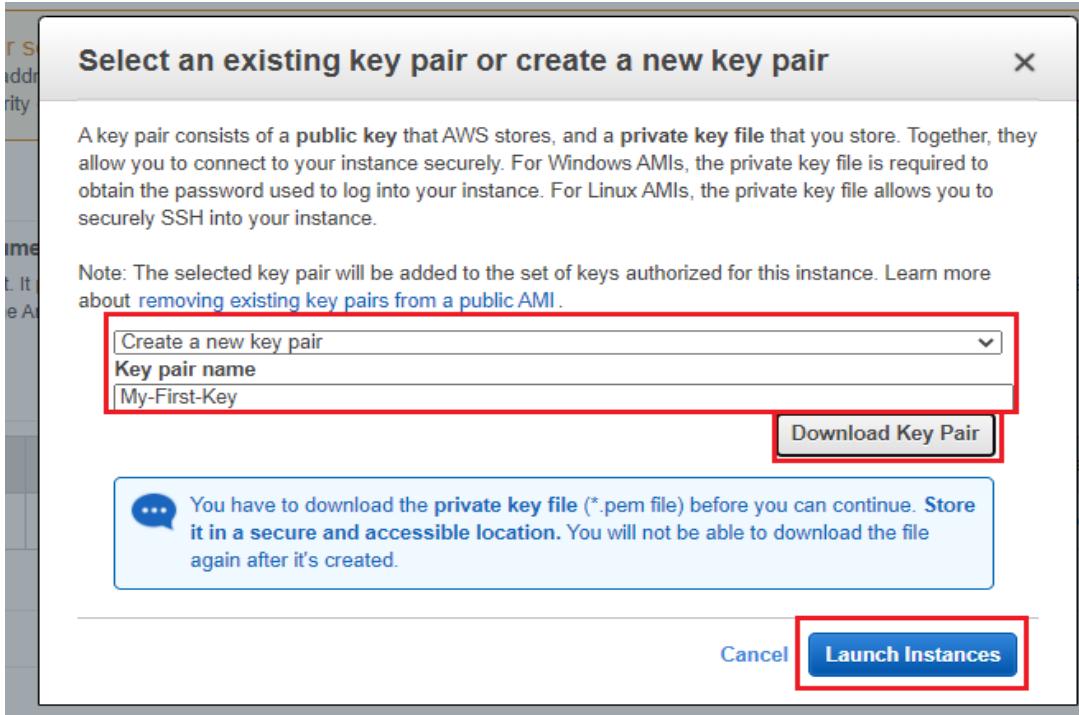
Security Groups

[Edit security groups](#)

Security group name: My-SG

Activate | Deactivate | [Cancel](#) | [Previous](#)**Launch**

5. Click on “Launch”.



5. Click on “Create a new key pair”.
6. Download Key Pair.
7. Click on “Launch Instance”.
8. Click on “View Instance”.

The screenshot shows the AWS EC2 Instances page. At the top, there's a header with 'Instances (1/3)' and 'Info' buttons, followed by 'Connect', 'Instance state', 'Actions', and 'Launch instances' buttons. Below the header is a search bar with 'Filter instances' placeholder text and navigation arrows. The main table lists three instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
First	i-0610ff454f83aca5b	Terminating	t2.micro	-	No alarms	ap-south-1a	-
Second	i-0da7e305183d1c90a	Terminating	t2.micro	-	No alarms	ap-south-1b	-
My-First-Inst...	i-091152db64fd104a5	Running	t2.micro	Initializing	No alarms	ap-south-1b	ec2-15

Below the table, a modal window is open for the selected instance 'My-First-Inst...'. The title is 'Instance: i-091152db64fd104a5 (My-First-Instance)'. The modal has tabs for 'Details', 'Security', 'Networking', 'Storage', 'Status checks', 'Monitoring', and 'Tags'. The 'Details' tab is active, showing sections for 'Instance summary' and 'Instance details'. The 'Platform' and 'AMI ID' fields are visible at the bottom.

9. Select your instance and check the Instance state, it must be "Running".

Congratulations!!!!!!

Steps to create EC2 Instance.



Step 2: Connect to your instance

There are several ways to connect to your Linux instance. For more information, see [Connect to your Linux instance](#).

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like New EC2 Experience, EC2 Dashboard, Events, Tags, Limits, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and Elastic Block Store. The main area displays three instances: First, Second, and My-First-Inst... (selected). The 'Connect' button in the top right and the 'Connect' option in the context menu for the selected instance are both highlighted with red boxes.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
First	i-06f0ff454f83aca5b	Terminated	t2.micro	-	No alarms	ap-south-1a	-
Second	i-0da7e305183d1c90a	Terminated	t2.micro	-	No alarms	ap-south-1b	-
My-First-Inst...	i-091152db64fd104a5	Running	t2.micro	2/2 checks ...	No alarms	ap-south-1b	ec2-15-

1. Click on “Connect”.
2. Or right click on Instance and select “Connect”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.

Step 2: Connect to your instance



Connect to instance Info

Connect to your instance i-091152db64fd104a5 (My-First-Instance) using any of these options

EC2 Instance Connect **Session Manager** **SSH client**

Instance ID
 [i-091152db64fd104a5 \(My-First-Instance\)](#)

Public IP address
 [15.206.212.136](#)

User name

Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.

 **Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel **Connect**

1. There are 3 ways to connect your instance.
2. Click on “EC2 Instance Connect”.
3. Click on “Connect”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.



Step 2: Connect to your instance

Connect to instance Info

Connect to your instance i-091152db64fd104a5 (My-First-Instance) using any of these options

EC2 Instance Connect **Session Manager** **SSH client**

Instance ID
 [i-091152db64fd104a5 \(My-First-Instance\)](#)

Public IP address
 [15.206.212.136](#)

User name

Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.

 **Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel **Connect**

1. There are 3 ways to connect your instance.
2. Click on “EC2 Instance Connect”.
3. Click on “Connect”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.



How to install Apache Web server on EC2 instance.

The screenshot shows a terminal window with the URL <https://ap-south-1.console.aws.amazon.com/ec2/v2/connect/ec2-user/i-091152db64fd104a5>. The terminal output shows:

```
Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-14-195 ~]$ sudo su
[root@ip-172-31-14-195 ec2-user]# yum update -y
```

The screenshot shows a terminal window with the URL <https://ap-south-1.console.aws.amazon.com/ec2/v2/connect/ec2-user/i-091152db64fd104a5>. The terminal output shows several blank lines followed by:

```
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]# sudo yum install -y httpd.x86_64
```

1. Give command -> sudo su and yum update -y
2. Sudo yum install -y httpd.x86_64 => to install Apache HTTP Server

Steps to create EC2 Instance.



How to install Apache Web server on EC2 instance.

```
← → C https://ap-south-1.console.aws.amazon.com/ec2/v2/connect/ec2-user/i-091152db64fd104a5
[ec2-user@ip-172-31-14-195 ~]$ 
[ec2-user@ip-172-31-14-195 ~]$ 
[ec2-user@ip-172-31-14-195 ~]$ 
[ec2-user@ip-172-31-14-195 ~]$ 
[ec2-user@ip-172-31-14-195 ~]$ sudo su
root@ip-172-31-14-195 ec2-user]# systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
root@ip-172-31-14-195 ec2-user]# systemctl start httpd
root@ip-172-31-14-195 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2021-02-14 12:57:54 UTC; 6s ago
     Docs: man:httpd.service(8)
 Main PID: 4950 (httpd)
   Status: "Processing requests..."
  CGroup: /system.slice/httpd.service
          ├─4950 /usr/sbin/httpd -DFOREGROUND
          ├─4951 /usr/sbin/httpd -DFOREGROUND
          ├─4952 /usr/sbin/httpd -DFOREGROUND
          ├─4953 /usr/sbin/httpd -DFOREGROUND
          ├─4954 /usr/sbin/httpd -DFOREGROUND
          └─4955 /usr/sbin/httpd -DFOREGROUND
Feb 14 12:57:54 ip-172-31-14-195.ap-south-1.compute.internal systemd[1]: Starting The Apache HTTP Server...
Feb 14 12:57:54 ip-172-31-14-195.ap-south-1.compute.internal systemd[1]: Started The Apache HTTP Server.
[root@ip-172-31-14-195 ec2-user]#
```

1. Give command
2. Systemctl enable httpd => to enable httpd service
3. Systemctl start httpd => to start httpd service
4. Systemstl status httpd => to check httpd service status
5. Curl localhost:80 => to check wheather apache server is responding on port 80

Steps to create EC2 Instance.



How to install Apache Web server on EC2 instance.

The screenshot shows a terminal window with the following session:

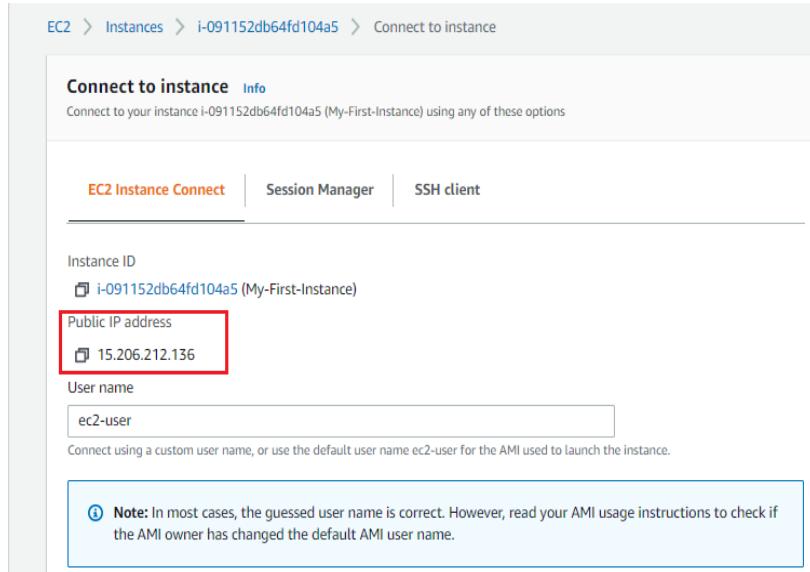
```
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]# hostname -f  
ip-172-31-14-195.ap-south-1.compute.internal  
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]# echo "My Name is Bond. James Bond 007. $(hostname -f)" > /var/www/html/index.html  
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]#
```

The command `echo "My Name is Bond. James Bond 007. $(hostname -f)" > /var/www/html/index.html` and its output are highlighted with a red box.

1. Give command
2. Hostname –f => to know name of your ec2 instance
3. Echo “My name is bond. James Bond 007. \$(hostname –f)” > /var/www/html/index.html

Steps to create EC2 Instance.

How to install Apache Web server on EC2 instance.



EC2 > Instances > i-091152db64fd104a5 > Connect to instance

Connect to instance Info

Connect to your instance i-091152db64fd104a5 (My-First-Instance) using any of these options

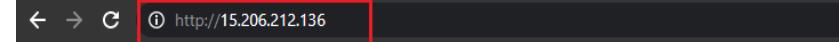
EC2 Instance Connect Session Manager SSH client

Instance ID: i-091152db64fd104a5 (My-First-Instance)

Public IP address: 15.206.212.136 (highlighted with a red box)

User name: ec2-user

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.



← → C ⓘ http://15.206.212.136



This site can't be reached

15.206.212.136 took too long to respond.

Try:

- Checking the connection
- Checking the proxy and the firewall
- Running Windows Network Diagnostics

ERR_CONNECTION_TIMED_OUT

1. Copy the public IP of ec2 instance and paste in browser.
2. You will get an error “ERR_CONNECTION_TIMED_OUT”

Lets Solve this issue now.....

Steps to create EC2 Instance.

How to solve connection error. Whenever there is timeout problem, check your “Security Groups”.



The screenshot shows the AWS Management Console interface for the Amazon EC2 service. On the left, a navigation menu lists various AWS services like Spot Requests, Savings Plans, Reserved Instances, etc., with 'Network & Security' expanded to show 'Security Groups'. The 'Security Groups' link is highlighted with a red box. The main content area displays a table of instances. One instance, 'My-First-Inst...', is selected and shown in detail below the table. The instance is labeled 'Running' and has an 'Instance type' of 't2.micro'. It is associated with a security group named 'My-SG'.

Click on “Security Groups” from left side menu. Our EC2 instance is attached with “My-SG” security group. Select that and click on “Actions” then select “Edit Inbound Rules”.

The screenshot shows the AWS Management Console interface for the Security Groups service. The left sidebar shows the same navigation menu as the previous screenshot. The main table lists four security groups. The third row, 'My-SG', is selected and highlighted with a red box. To the right of the table, a context menu is open under the 'Actions' button, also highlighted with a red box. The 'Edit inbound rules' option is clearly visible within this menu.

Steps to create EC2 Instance.

How to solve connection error. Whenever there is timeout problem, check your “Security Groups”.



The screenshot shows the AWS Security Groups Inbound rules configuration page. It displays two rules:

- Rule 1:** Type: SSH, Protocol: TCP, Port range: 22, Source: Custom (0.0.0.0/0), Description: allow traffic from all IP addresses.
- Rule 2:** Type: HTTP, Protocol: TCP, Port range: 80, Source: Custom (0.0.0.0/0), Description: allow traffic from anywhere on port 80.

A red box highlights the "Add rule" button at the bottom left. Another red box highlights the second rule's source field. A note at the bottom states: "⚠ NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created." At the bottom right, there are "Cancel", "Preview changes", and "Save rules" buttons, with "Save rules" highlighted by a red box.

Click on “Add Rule” . Select “HTTP”. And then click on “Save rules”.

Now, again go on browser and paste public IP and hit enter. You will get the output.

The screenshot shows a browser window with the address bar displaying "Not secure | http://15.206.212.136". The main content area shows the text "My Name is Bond. James Bond 007. ip-172-31-14-195.ap-south-1.compute.internal", which is highlighted by a red box.

Congratulations!!!!!!

Steps to create EC2 Instance.



Step 3: Clean up your instance

- 1) After you've finished with the instance that you created for this tutorial, you should clean up by terminating the instance.
- 2) If you launched an instance that is not within the [AWS Free Tier](#), you'll stop incurring charges for that instance as soon as the instance status changes to shutting down or terminated.

The screenshot shows the AWS EC2 Instances page. A single instance named "My-First-Inst..." is listed. The "Actions" menu for this instance is open, showing options: Stop instance, Start instance, Reboot instance, Hibernate instance, and Terminate instance. The "Terminate instance" option is highlighted with a red box. On the right side of the screen, the instance details panel shows the instance state as "t2.micro" and other attributes like "arm status", "Availability Zone", and "Public IP".

Click on "Instance State" . Or right click on your "Instance Id". Select "Stop instance" or "Terminate instance".

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.



Step 3: Clean up your instance

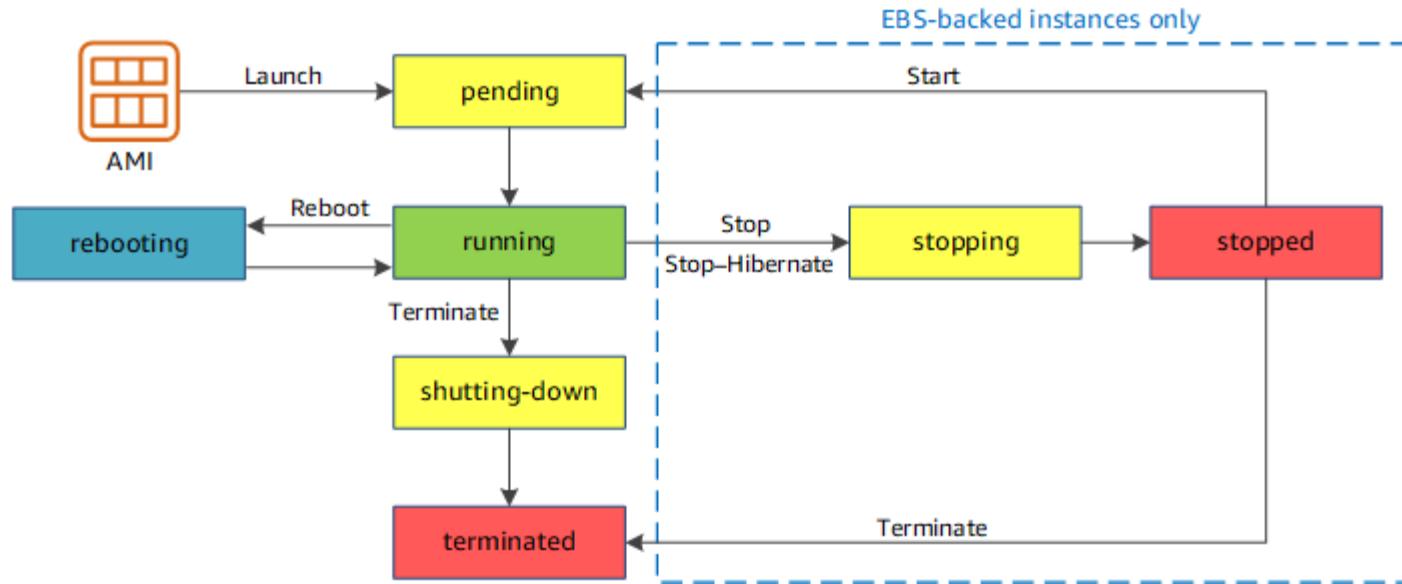
- 1) After you've finished with the instance that you created for this tutorial, you should clean up by terminating the instance.
- 2) If you launched an instance that is not within the [AWS Free Tier](#), you'll stop incurring charges for that instance as soon as the instance status changes to shutting down or terminated.

The screenshot shows the AWS EC2 Instances page. A single instance, "My-First-Inst...", is listed. A context menu is open over this instance, showing options like "Launch instances", "Launch instance from template", "Connect", and four options highlighted with red boxes: "Stop instance", "Start instance", "Reboot instance", and "Terminate instance". Another context menu is open over the "Instance state" column header, also showing "Stop instance", "Start instance", "Reboot instance", and "Terminate instance" options highlighted with red boxes. The instance details panel on the left shows the instance ID "i-091152db64fd1015", the instance type "t2.micro", and the owner information "Owner 933126026785".

Click on “Instance State” . Or right click on your “Instance Id”. Select “Stop instance” or “Terminate instance”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Instance lifecycle



Ref Link: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-lifecycle.html>

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- IAM Introduction
- Compute services (EC2)
- **Storage services (S3 Buckets, EBS)**
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)

S3 (Simple Storage Service)



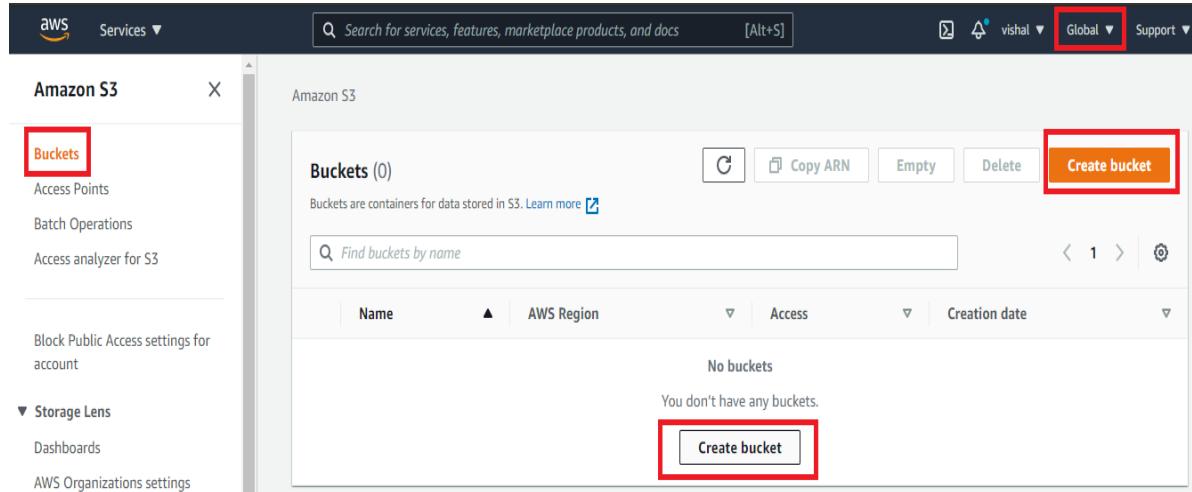
- ❖ Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- ❖ You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- ❖ Amazon S3 stores data as objects within buckets.
- ❖ An object is a file and any optional metadata that describes the file.
- ❖ To store a file in Amazon S3, you upload it to a bucket.
- ❖ When you upload a file as an object, you can set permissions on the object and any metadata.
- ❖ Buckets are containers for objects. You can have one or more buckets.
- ❖ You can control access for each bucket, deciding who can create, delete, and list objects in it.
- ❖ You can also choose the geographical Region where Amazon S3 will store the bucket and its contents and view access logs for the bucket and its objects.

Ref link: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

Use Case: Deploy your own static website on S3

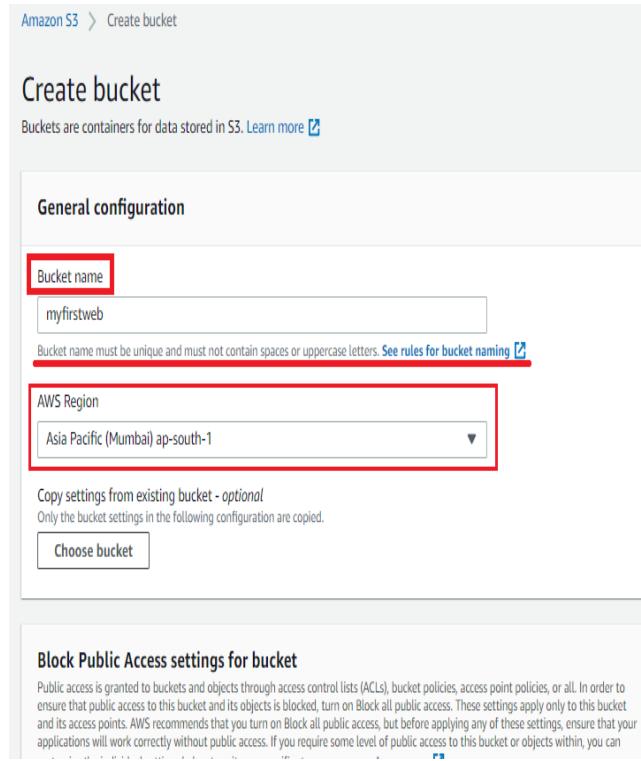
Steps:

- Step 1: Create a sample web application on your local machine.
- Step 2: Login to AWS console
- Step 3: Search for S3 service



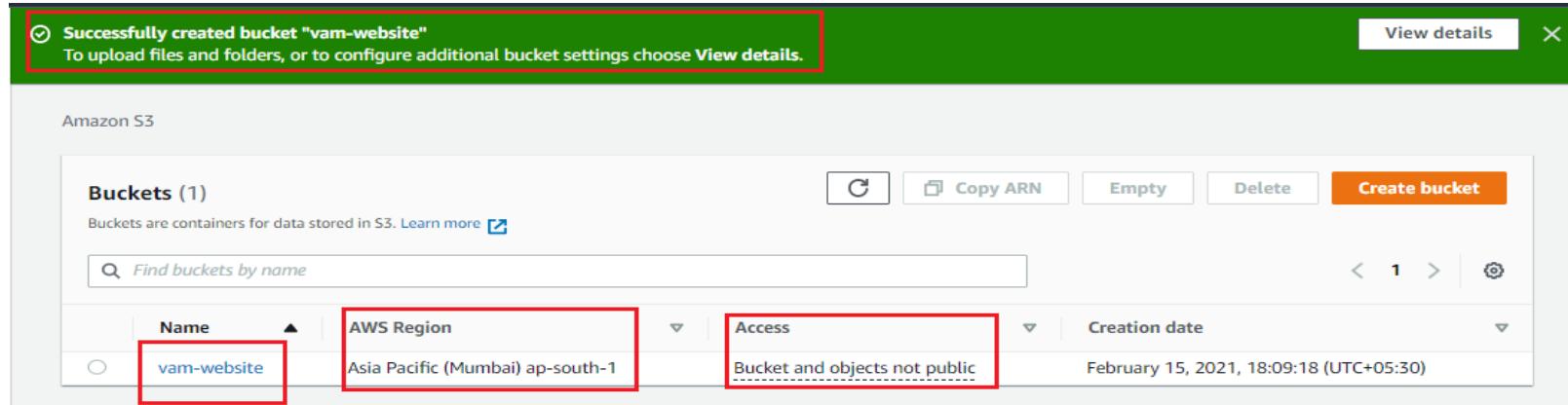
The screenshot shows the AWS S3 service page. On the left, there's a sidebar with options like 'Buckets' (which is selected and highlighted with a red box), 'Access Points', 'Batch Operations', 'Access analyzer for S3', 'Block Public Access settings for account', 'Storage Lens', 'Dashboards', and 'AWS Organizations settings'. The main area is titled 'Amazon S3' and shows 'Buckets (0)'. It includes a search bar, a 'Create bucket' button (highlighted with a red box), and a 'Create bucket' link below it. A message says 'No buckets' and 'You don't have any buckets.' There's also a 'Create bucket' button at the bottom of the list.

- Step 4: Click on “Create Bucket”
- Step 5: give “Bucket Name” which must be globally unique.
- Step 6: select region
- Step 7: Click on “Create Bucket”



The screenshot shows the 'Create bucket' wizard. The first step is 'General configuration'. It has a 'Bucket name' field containing 'myfirstweb' (highlighted with a red box) and an 'AWS Region' dropdown set to 'Asia Pacific (Mumbai) ap-south-1' (highlighted with a red box). Below these, there's a note about copying settings from an existing bucket and a 'Choose bucket' button. At the bottom, there's a section for 'Block Public Access settings for bucket' with a note about public access controls.

Use Case: Deploy your own static website on S3



Successfully created bucket "vam-website". To upload files and folders, or to configure additional bucket settings choose [View details](#).

View details X

Amazon S3

Buckets (1)

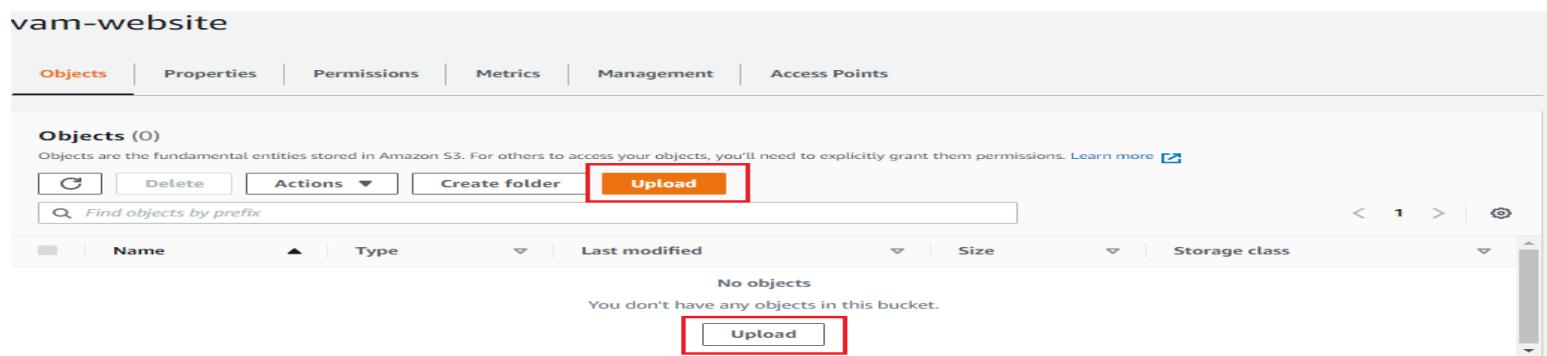
Buckets are containers for data stored in S3. [Learn more](#)

Find buckets by name

Name	AWS Region	Access	Creation date
vam-website	Asia Pacific (Mumbai) ap-south-1	Bucket and objects not public	February 15, 2021, 18:09:18 (UTC+05:30)

Congratulations!!!! - A bucket is successfully created.

Step 8: Right click on Bucket Name. and click on “Upload”



vam-website

Objects Properties Permissions Metrics Management Access Points

Objects (0)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Actions Upload

Find objects by prefix

Name	Type	Last modified	Size	Storage class
No objects				

You don't have any objects in this bucket.

Upload

Use Case: Deploy your own static website on S3

Upload

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.

Files and folders (0)					
All files and folders in this table will be uploaded.					
<input type="text"/> Find by name					
Name	Folder	Type	Size		
No files or folders					
You have not chosen any files or folders to upload.					

[Remove](#) [Add files](#) [Add folder](#)

Step 9: Click on “Add Folder”. Select your website folder and upload it. And click on “Upload” Button.

Step 10: go on your Object Page.

Upload

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.

Files and folders (3 Total, 862.0 B)					
All files and folders in this table will be uploaded.					
<input type="text"/> Find by name					
Name	Folder	Type	Size		
error.html	mywebsite/	text/html	269.0 B		
index.html	mywebsite/	text/html	312.0 B		
page2.html	mywebsite/	text/html	281.0 B		

[Remove](#) [Add files](#) [Add folder](#)

Destination

Destination
<s3://vam-website>

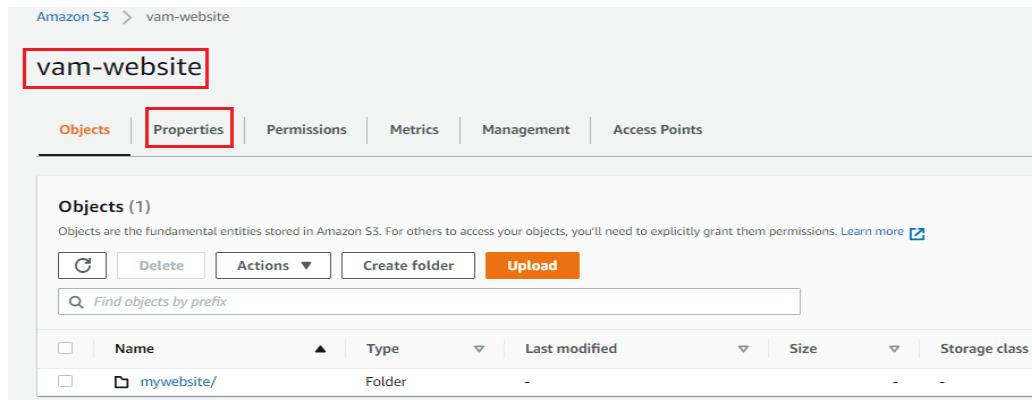
Use Case: Deploy your own static website on S3

Step 9: Click on “Add Folder”. Select your website folder and upload it. And click on “Upload” Button.

Step 10: go on your Object Page.

Step 11: Click on “Properties”

Step 12: At bottom “Static Website Hosting” click on “Edit”



Amazon S3 > vam-website

vam-website

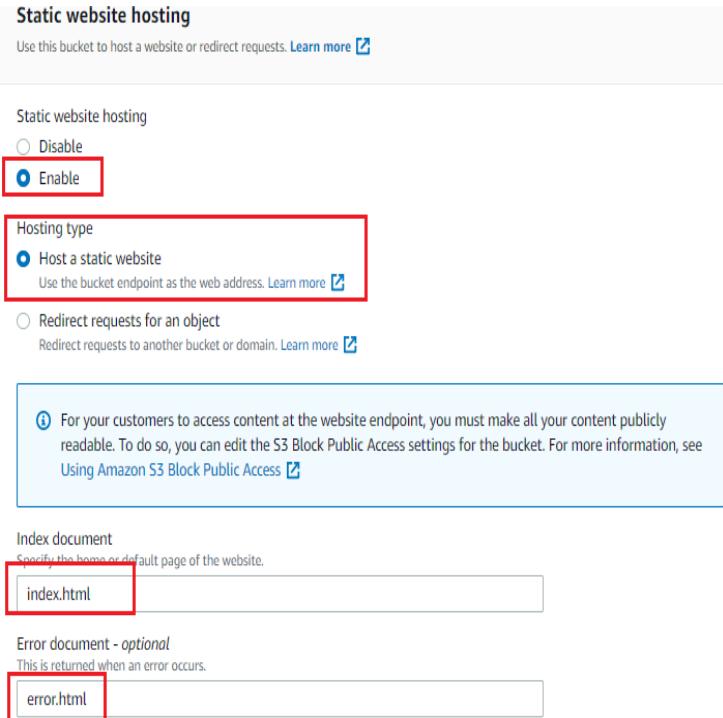
Objects (1)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Name	Type	Last modified	Size	Storage class
mywebsite/	Folder	-	-	-

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)



Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Disable

Enable

Hosting type

Host a static website

Use the bucket endpoint as the web address. [Learn more](#)

Redirect requests for an object

Redirect requests to another bucket or domain. [Learn more](#)

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

Index document

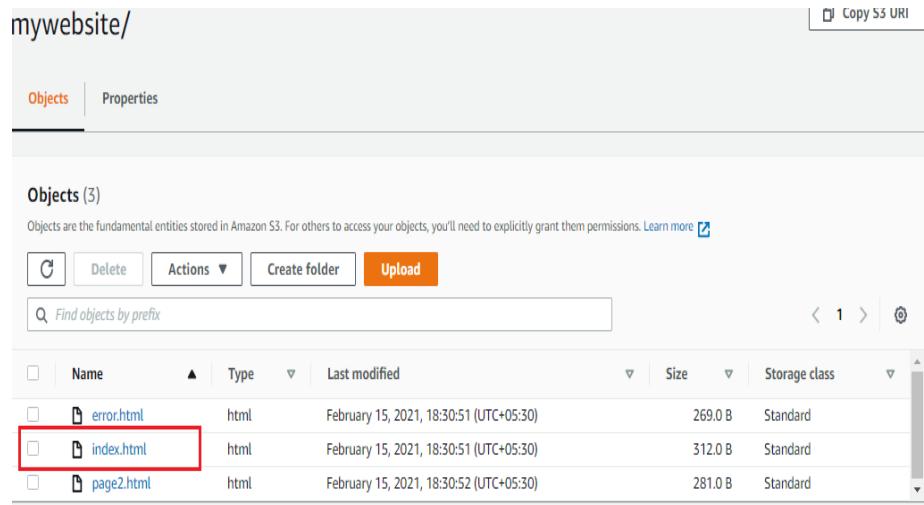
Specify the home or default page of the website.

Error document - optional

This is returned when an error occurs.

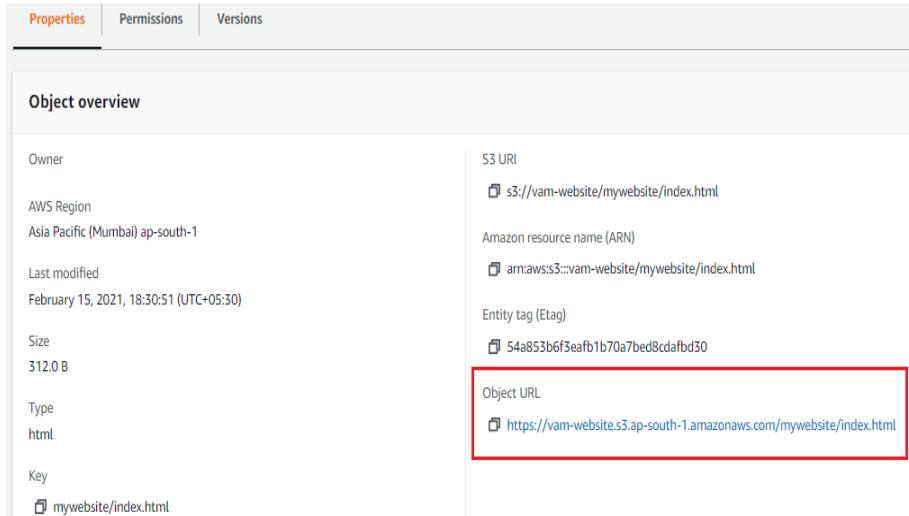
Use Case: Deploy your own static website on S3

Step 12: Click on “index.html” object. And get the object URL



The screenshot shows the AWS S3 console interface. At the top, there's a search bar with 'mywebsite/' and a 'Copy S3 URI' button. Below it, tabs for 'Objects' and 'Properties' are visible. Under 'Objects (3)', a table lists three files: 'error.html', 'index.html', and 'page2.html'. The 'index.html' row is selected and highlighted with a red box. The table columns include Name, Type, Last modified, Size, and Storage class. At the bottom of the table are buttons for 'Delete', 'Actions ▾', 'Create folder', and 'Upload'.

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	error.html	html	February 15, 2021, 18:30:51 (UTC+05:30)	269.0 B	Standard
<input checked="" type="checkbox"/>	index.html	html	February 15, 2021, 18:30:51 (UTC+05:30)	312.0 B	Standard
<input type="checkbox"/>	page2.html	html	February 15, 2021, 18:30:52 (UTC+05:30)	281.0 B	Standard



The screenshot shows the 'Object overview' page for the 'index.html' object. It displays various metadata fields: Owner, AWS Region (Asia Pacific (Mumbai) ap-south-1), Last modified (February 15, 2021, 18:30:51 (UTC+05:30)), Size (312.0 B), Type (html), and Key (mywebsite/index.html). On the right side, there are fields for S3 URI, Amazon resource name (ARN), Entity tag (Etag), and a large 'Object URL' field containing the value <https://vam-website.s3.ap-south-1.amazonaws.com/mywebsite/index.html>, which is also highlighted with a red box.

Step 13: Copy the URL in browser and you will get an Error “Access Denied”.



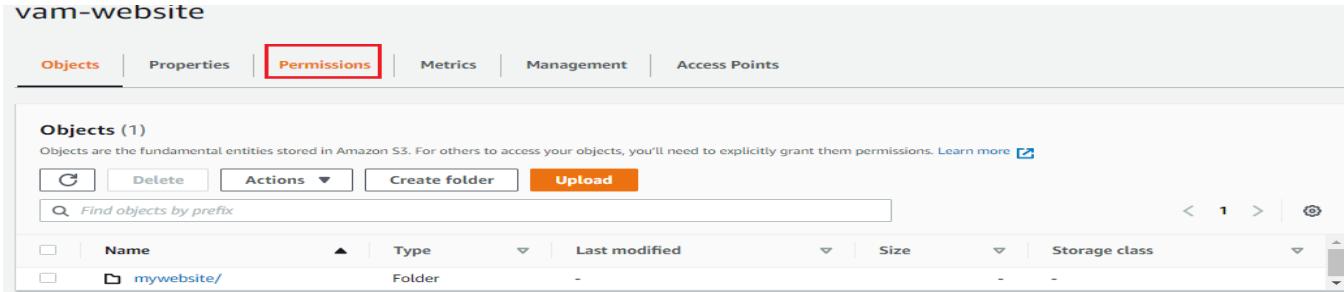
The screenshot shows a browser window with the URL <https://vam-website.s3.ap-south-1.amazonaws.com/mywebsite/index.html>. The page content is an XML error response:

```
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<RequestId>D9FF1B7487C054F0</RequestId>
<HostId>M5z/VKfygnJPVXlkqnqS2nXKdMY/RxpjnkZaF3eTja9p9COOXeF3hT5cDGo6Tf1Uj7xbxFWh4Q=</HostId>
```

Lets Solve this issue now.

Use Case: Deploy your own static website on S3

Step 14: Visit your bucket page and click on “Permission”



The screenshot shows the AWS S3 console for a bucket named 'vam-website'. The 'Permissions' tab is highlighted with a red border. Under the 'Objects' section, there is one object named 'mywebsite/'. The object type is listed as 'Folder'.

Step 15: Click on “Bucket Public access - Edit” . Uncheck the “Block all public access” box. Click on “Save Changes”

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

On

- Block public access to buckets and objects granted through new access control lists (ACLs)
 On
- Block public access to buckets and objects granted through any access control lists (ACLs)
 On
- Block public access to buckets and objects granted through new public bucket or access point policies
 On
- Block public and cross-account access to buckets and objects through any public bucket or access point policies
 On

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

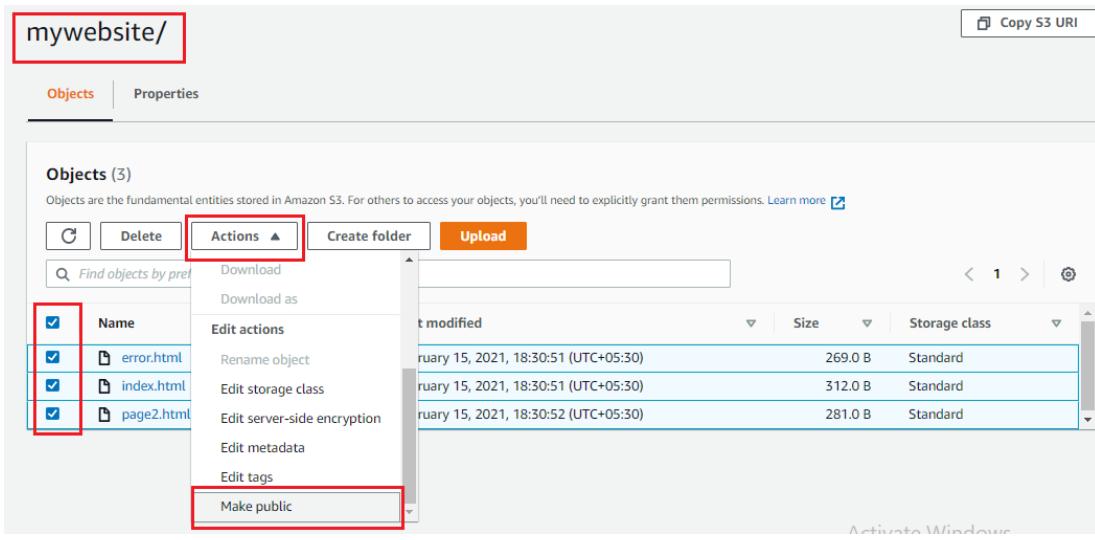
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- **Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Use Case: Deploy your own static website on S3

Step 16: again go to object page, copy “index.html” link and open in browser. If still problem remain then do the following steps:

Step 17: open Object page. Select all “Object” click on “Action” and select “Make Public”.



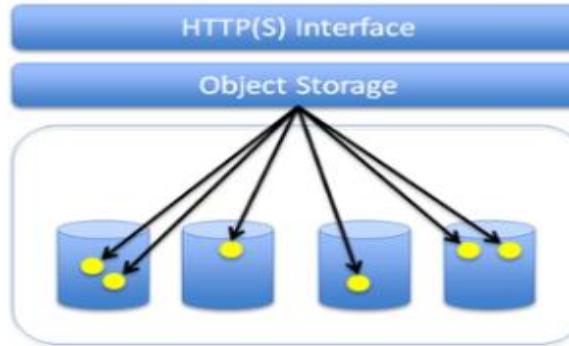
The screenshot shows the AWS S3 console with the path "mywebsite/" selected in the left sidebar. The main area displays three objects: "error.html", "index.html", and "page2.html". The "Actions" button is highlighted with a red box. A dropdown menu is open, showing options like "Edit actions", "Rename object", "Edit storage class", etc., with "Make public" at the bottom also highlighted with a red box.



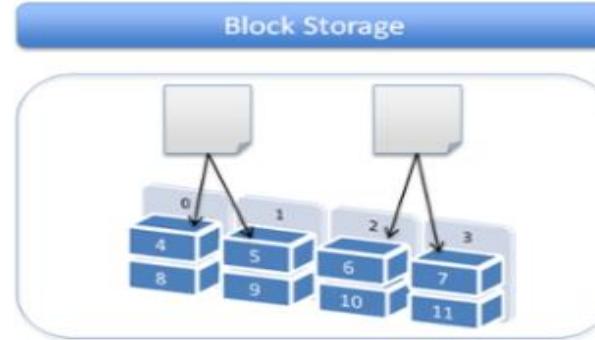
Step 18: Now copy the index.html url and open in browser.

Congratulations!!!!!!

Difference between S3, EBS and EFS



- Store virtually unlimited files.
- Maintain file revisions.
- HTTP(S) based interface.
- Files are distributed in different physical nodes.



- File is split and stored in fixed sized blocks.
- Capacity can be increased by adding more nodes.
- Suitable for applications which require high IOPS, database, transactional data.

Image Source: [NetApp Cloud](#) (used with permission)

Difference between S3, EBS and EFS

Category	S3	EBS	EFS
Storage Type	Object Storage	Block Storage	File Storage
Pricing	Pay as you Use	Pay for provisioned capacity	Pay as you Use
Storage Size	Unlimited Storage	Limited storage	Unlimited Storage
Scalability	Unlimited Scalability	Increase/decrease size manually	Unlimited Scalability
Durability	Stored redundantly across multiple Azs	Stored redundantly in a Single AZ	Stored redundantly across multiple Azs
Availability	Max is 99.99% with S3	99.99%	No SLAs
Security	Supports Data at Rest and Data in Transit encryption	Supports Data at Rest and Data in Transit encryption	Supports Data at Rest and Data in Transit encryption
Back up and Restore	Use Versioning or cross-region replication	Automated Backups and Snapshots	EFS to EFS replication
Performance	Slower than EBS and EFS	Faster than S3 and EFS	Faster than S3, Slower than EBS
Accessibility	Publicly and Privately accessible	Accessible only via the attached EC2 instance	Accessible simultaneously from multiple EC2 and on-premises instances
Interface	Web Interface	File System Interface	Web and File System Interface
Use cases	Media, Entertainment, Big data analytics, backups and archives, web serving and content management	Boot volumes, transactional and NoSQL databases, data warehousing ETL	Media, Entertainment, Big data analytics, backups and archives, web serving and content management, home directories

Ref: <https://jayendrapatil.com/aws-s3-vs-ebs-vs-efs/>

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- IAM Introduction
- Compute services (EC2)
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)
- Lambda

Thank You

Presentation Topic

Cloud Computing

Vishal Ambadas Meshram
vishal.meshram@viit.ac.in

Department of Computer Engineering



BRACT'S, Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)

UNIT 1:

Introduction To Cloud Computing

Reference:

- 1) Thomas Erl, Zaigham Mahmood and Ricardo Puttini, “*Cloud Computing: Concepts, Technology and Architecture*”,
- 2) Wikipedia
- 3) Edureka

Contents

- Overview, Applications, Intranets and the Cloud.
- Your Organization and Cloud Computing- Benefits, Limitations, Security Concerns.
- Software as a Service (SaaS)- Understanding the Multitenant Nature of SaaS Solutions,
- Understanding SOA.
- Platform as a Service (PaaS).
- Infrastructure as a Service (IaaS)
- Case Study: Google Cloud Platform

CO Achieved in First Lecture

- 1) To understand cloud computing concepts
- 2) To study supporting technologies of cloud

Course Outcomes:-

At the end of the unit you will be able to:

- 1) Summarize the basic concepts of cloud computing
(Understand)
- 2) Make use of supporting technologies for cloud computing
(Understand, Apply)

University Questions

Q1)

- a) Comment on “Cloud and virtualization”
- b) What do you mean by pods, aggregation and silos in terms of IAAS
- c) Enlist and explain the challenges and obstacles while adopting cloud computing?

OR

Q2)

- a) What is administrating and monitoring cloud services? Enlist and explain different tools used for administrating and monitoring cloud services.
- b) Enlist services provided by SAAS, PAAS and IAAS
- c) How do you relate cloud computing with utility computing?

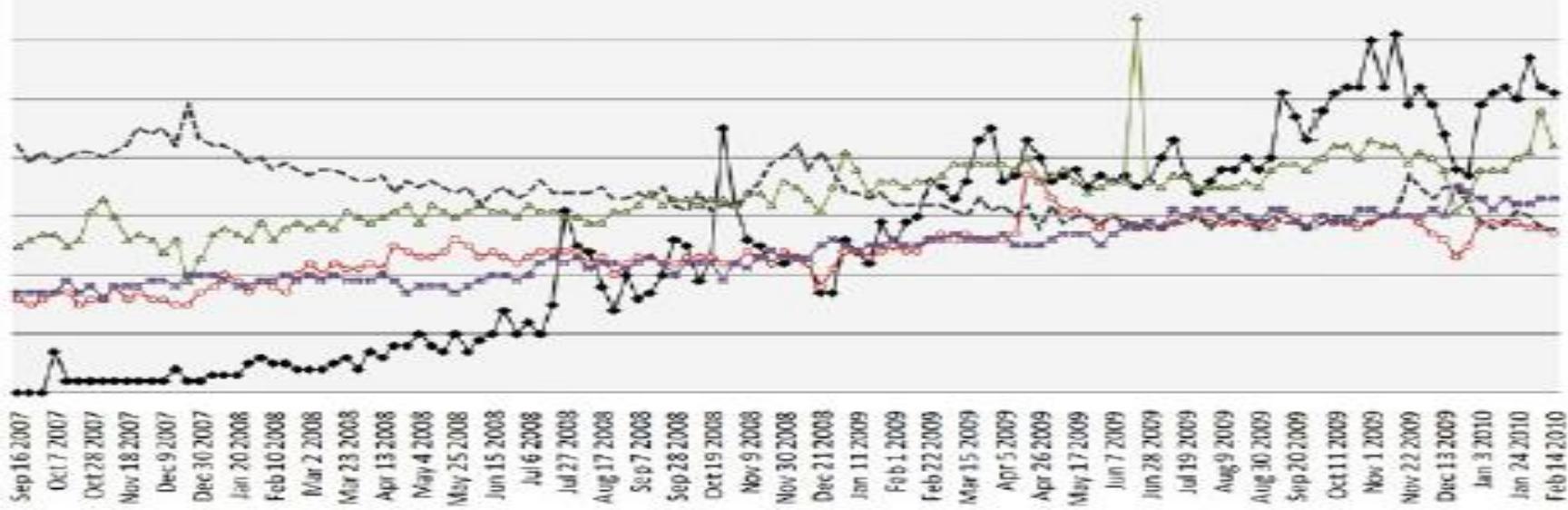
Popularity of cloud computing

Top 10 Strategic Technology Areas for 2010 (Powered By Gartner)

- 1. Cloud Computing
- 2. Advanced Analytics
- 3. Client Computing
- 4. IT for Green
- 5. Reshaping the Data Center
- 6. Social Computing
- 7. Security – Activity Monitoring
- 8. Flash Memory
- 9. Virtualization for Availability
- 10. Mobile Applications

Top Strategic Technologies (Powered By Google Trends)

—●— Cloud Computing —●— GreenIT —●— Social Network —●— Flash Memory —●— Mobile Application



A brief history

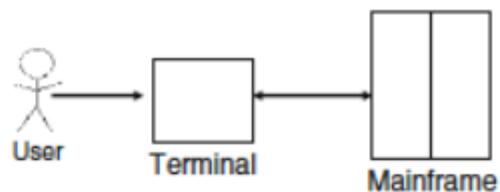
- The idea of computing in a “cloud” traces back to the origins of utility computing, a concept that computer scientists **John McCarthy** publically proposed in **1961**:

“ If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility..... the computer utility could become the basis of a new and important industry”

History

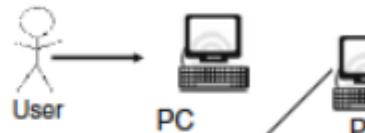
Many users shared powerful mainframes using dummy terminals

1. Mainframe Computing



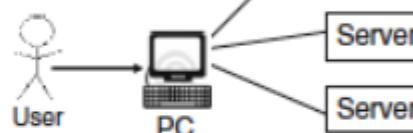
Stand-alone PCs became powerful enough to meet the majority of users' needs

2. PC Computing



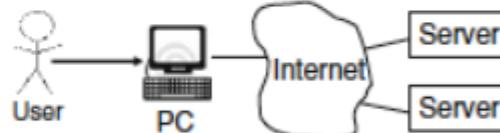
PCs, laptops, and servers were connected together through local networks to share resources and increase performance

3. Network Computing



Local networks were connected to other local networks forming a global network such as the Internet to utilise remote applications and resources

4. Internet Computing



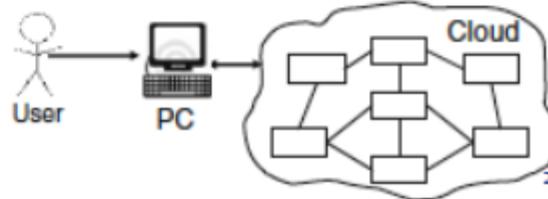
Grid computing provided shared computing power and storage through a distributed computing system

5. Grid Computing



Cloud computing further provides shared resources on the Internet in a scalable and simple way

6. Cloud Computing



Borko Furht, "Handbook of Cloud Computing"

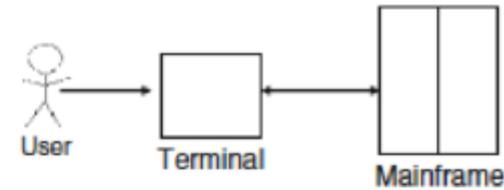
29

History

Offers finite computing power

Dummy terminals acted as user interface devices

1. Mainframe Computing

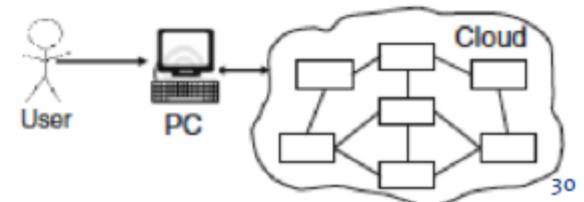


Quite similar?

Provides almost infinite power and capacity

Powerful PCs can provide local computing power and caching support

6. Cloud Computing



Borko Furht, "Handbook of Cloud Computing"

30

Introduction to Cloud Computing

- **Cloud Computing can be defined as**

“a new style of computing in which dynamically scalable and often virtualized resources are provided as a services over the Internet”.

Forrester Research provided its own definition of cloud computing:

“.... A standardized IT capability (services, software, or infrastructure) delivered via internet technologies in a pay-per-use, self-service way.”

Introduction to Cloud Computing

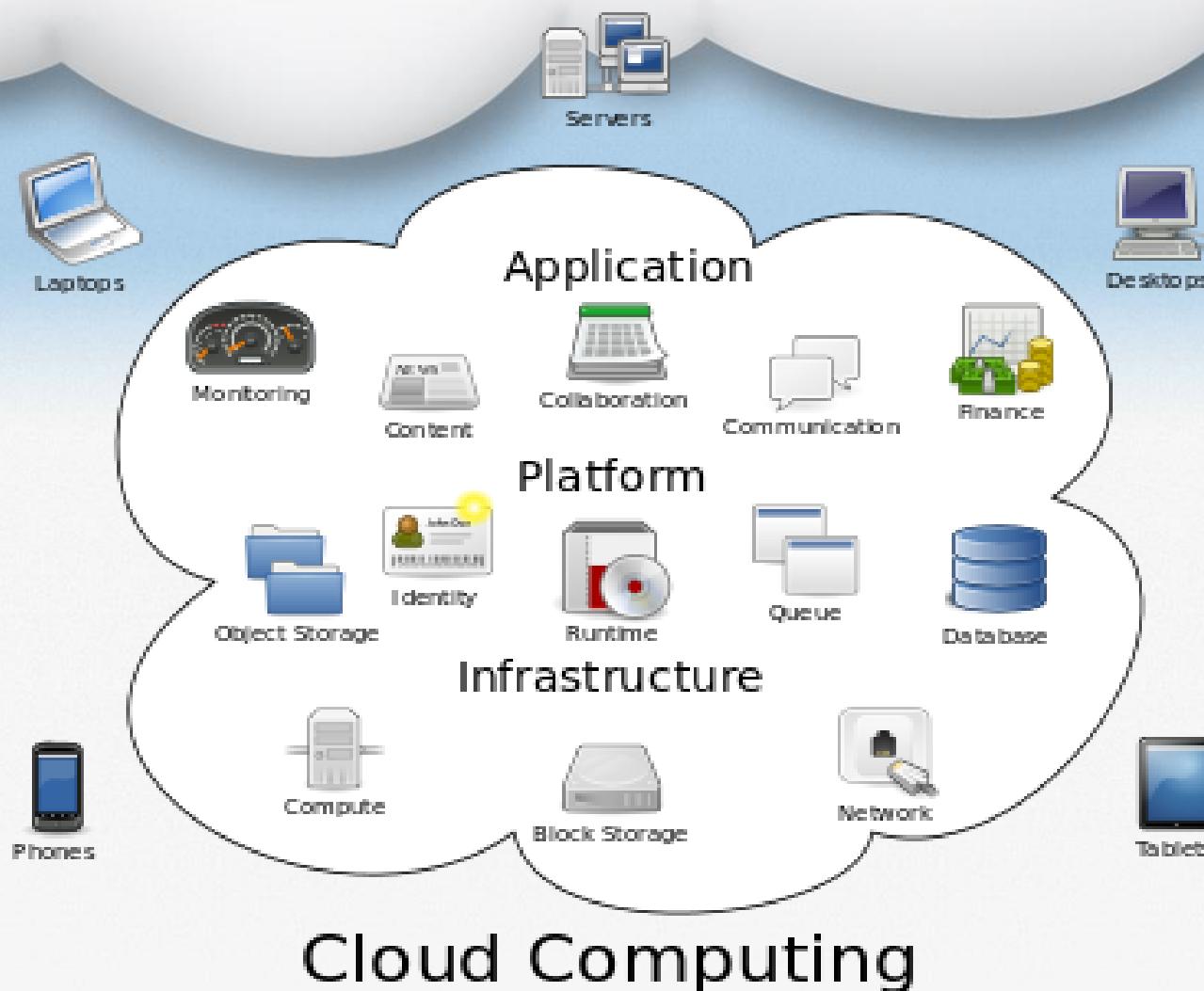
- **NIST (National Institute of Standards & Technology) Definition:**

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. network, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models. ”

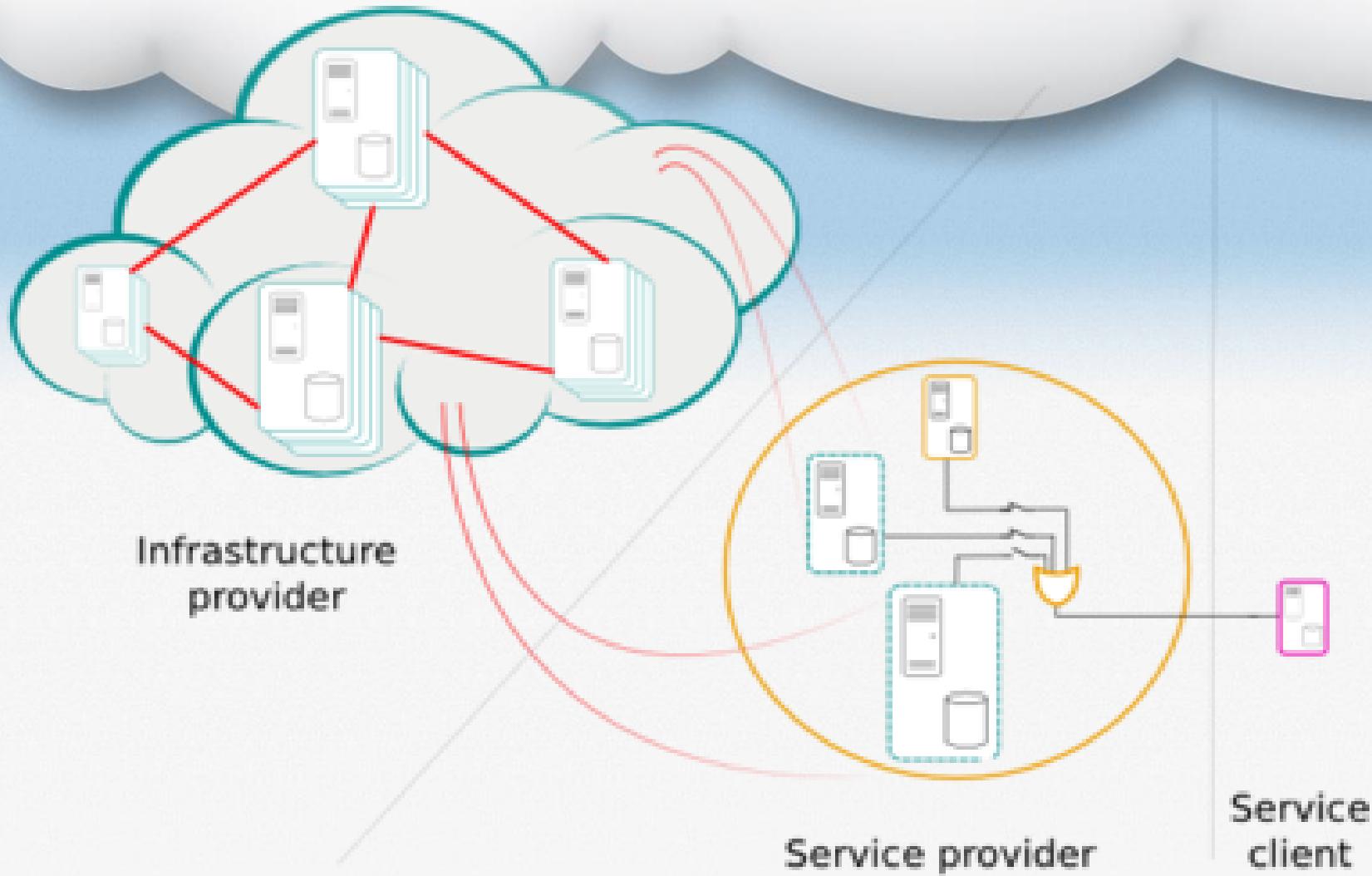
- A more concise definition:

“Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured resources”

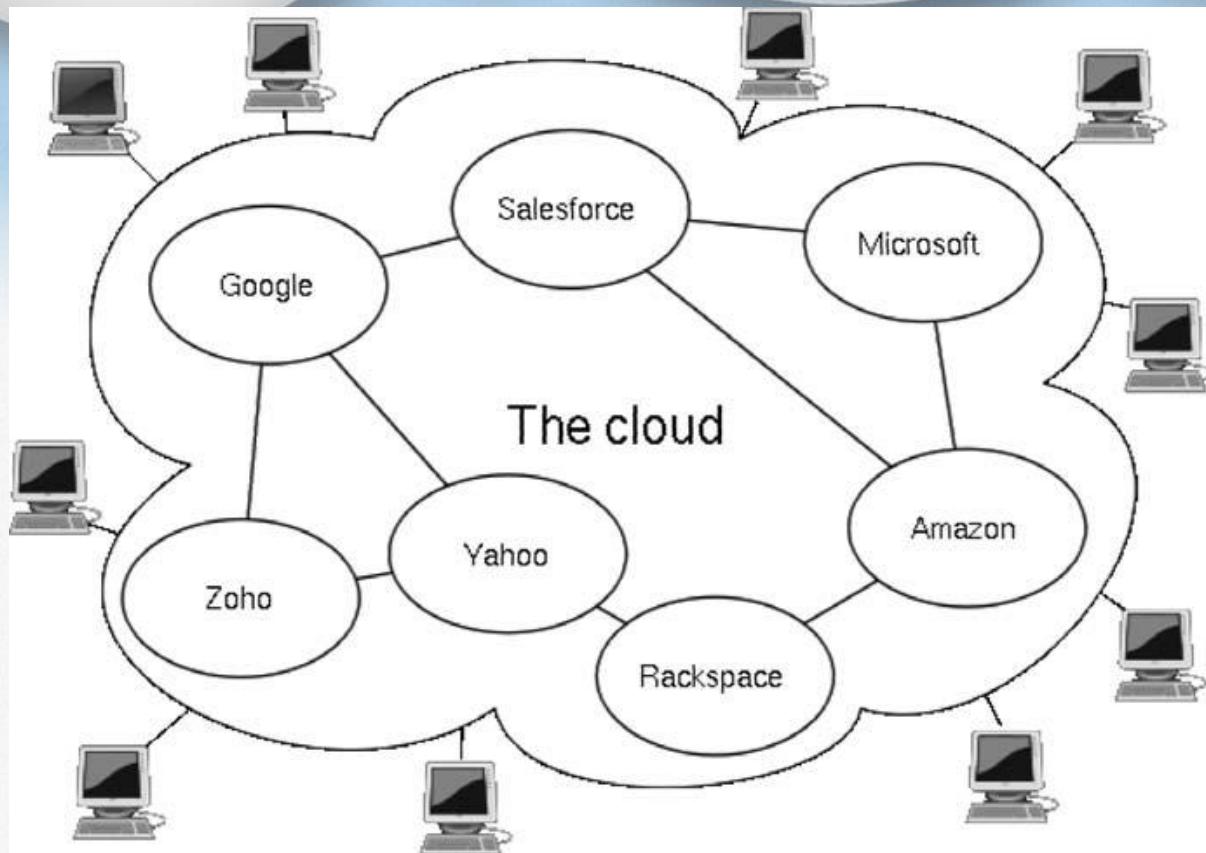
What is cloud computing?



What is cloud computing?



What is cloud computing?



Business Drivers

- Capacity Planning
- Cost Reduction
- Organizational Agility

Technology Innovations

- Clustering:

A cluster is a group of independent IT resources that are interconnected and work as a single system. System failure rates are reduced while availability and reliability are increased, since redundancy and failover are inherent to the cluster.

A general prerequisite of hardware clustering is that its component systems have reasonably identical hardware and operating systems to provide similar performance levels when one failed component is to be replaced by another. Component devices that form a cluster are kept in synchronization through dedication, high-speed communication links.

The basic concept of built-in redundancy and failover is core to cloud platforms.

Technology Innovations

- Grid Computing:

A computing grid (or “Computational grid”) provides a platform in which computing resources are organized into one or more logical pools. These pools are collectively coordinated to provide a high performance distributed grid, sometimes referred to as a “super virtual computer”.

Grid computing differs from clustering in that grid systems are much loosely coupled and distributed. As a result, grid computing systems can involve computing resources that are heterogeneous and geographically dispersed, which is generally not possible with cluster-based systems.

The technological advancements achieved by grid computing projects have influenced various aspects of cloud computing platforms and mechanism, specifically in relation to common feature- sets such as networked access, resource pooling, and scalability and resiliency.

These types of features can be established by both grid computing and cloud computing, in their own distinctive approaches.

Technology Innovations

- Virtualization:

Virtualization represents a technology platform used for the creation of virtual instances of IT resources.

A layer of virtualization software allows physical IT resources to provide multiple virtual images of themselves so that their underlying processing capabilities can be shared by multiple users.

Prior to the virtualization technology, software was limited to residing on and being coupled with static hardware environments. The virtualization process servers this software-hardware dependency, as hardware requirements can be simulated by emulation software running in virtualized environments.

Established virtualization technologies can be traced to several cloud characteristics and cloud computing mechanisms, having inspired many of their core features.

Modern virtualization technologies emerged to overcome the performance, reliability and scalability limitations of traditional virtualization platforms.

Technology Innovations Vs. Enabling Technologies

- It is essential to highlight several other areas of technology that continue to contribute to modern-day cloud-based platforms. There are distinguished as cloud-enabling technologies :
 - Broadband Networks and internet architecture
 - data center technology
 - (Modern) virtualization technology
 - Web technology
 - Multitenant technology
 - Service technology

Each of these cloud-enabling technologies existed in some form prior to the formal advent of cloud computing some were refined further, and on occasion even redefined, as a result of the subsequent evolution of cloud computing.

Summary of key points

- The primary business drivers that exposed the need of cloud computing and led to its formation include capacity planning, cost reduction, and organizational agility.
- The primary technology innovations that influenced and inspired key distinguishing features and aspects of cloud computing include clustering, grid computing, and traditional forms of virtualization.

Home Assignment Question

- Difference between Cluster, Grid and cloud computing??

- Tomorrow's Points:
 - Five essential characteristics,
 - Three service models, and
 - Four deployment models.

Muddiest Point

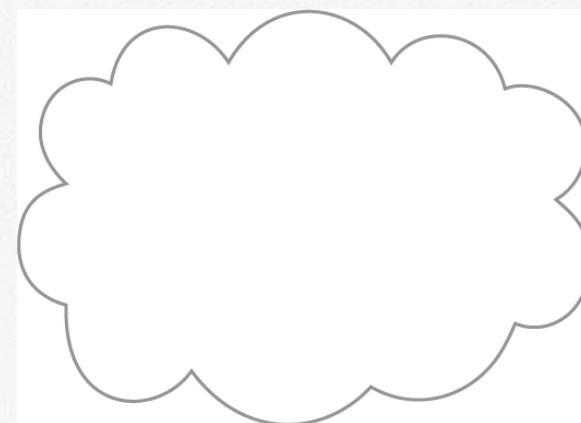
Thank You!!!!

Basic Concepts & Terminology

Cloud:

- A cloud refers to a distinct IT environment that is designed for the purpose of remotely provisioning scalable and measured IT resources.
- the symbol of cloud was commonly used to represent the Internet in a verity of specifications.
- This same symbol is now used to specifically represent the boundary of cloud environment, as shown in fig.

Fig: The symbol used to denote the boundary of a cloud environment.



Copyright © Arcitura Education

Basic Concepts & Terminology

Difference between “Cloud” and Internet :

1. As a specific environment used to remotely provision IT resources, a cloud has a finite boundary.
whereas the Internet provides open access to many Web-based IT resources, a cloud is typically privately owned and offers access to IT resources that is metered.

2. Much of the internet is dedicated to the access of content-based IT resources published via the World Wide Web,
on the other hand, are dedicated to supplying back-end processing capabilities and user-based access to these capabilities.

Basic Concepts & Terminology

□ IT Resources:

- An IT resource is a physical or virtual IT-related artifact that can be either software-based, such as virtual server or a custom software program, or hardware-based, such as a physical server or a network device.

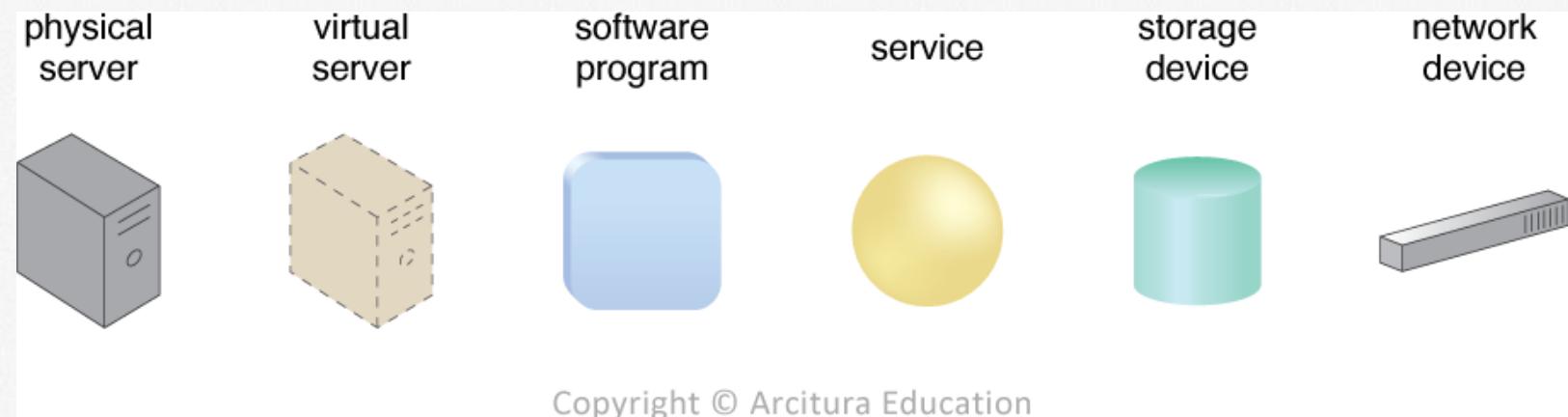


Figure: Examples of common IT resources and their corresponding symbols

Basic Concepts & Terminology

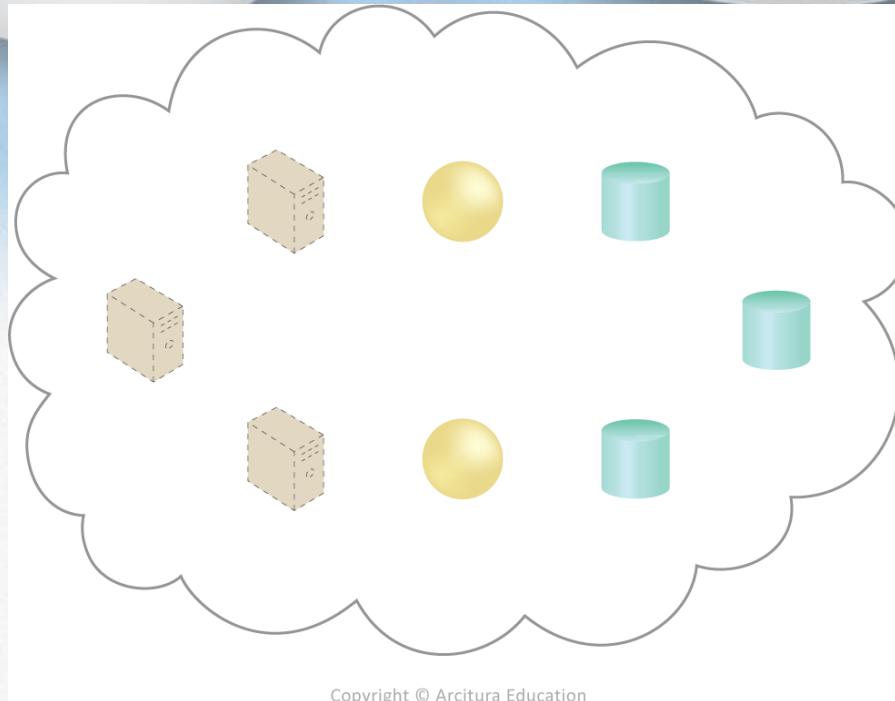


Figure: A cloud is hosting eight IT resources: three virtual servers, two cloud services, and three storage devices.

Basic Concepts & Terminology

On-Premise:

- As a distinct and remotely accessible environment, a cloud represents an option for the deployment of IT resources.
- An IT resources that hosted in a conventional IT enterprise within an organizational boundary (that does not specifically represent a cloud) is considered to be located on the premises of the IT enterprise, or on-premise for short.
- ***“on the premises of a controlled IT environment that is not cloud-based”***
- ***An IT resources that is on-premise cannot be a cloud-based, and vice-versa.***

Basic Concepts & Terminology

Scaling:

Scaling, from an IT recourse perspective, represents the ability of the IT resource to handle increased or decreased usage demands.

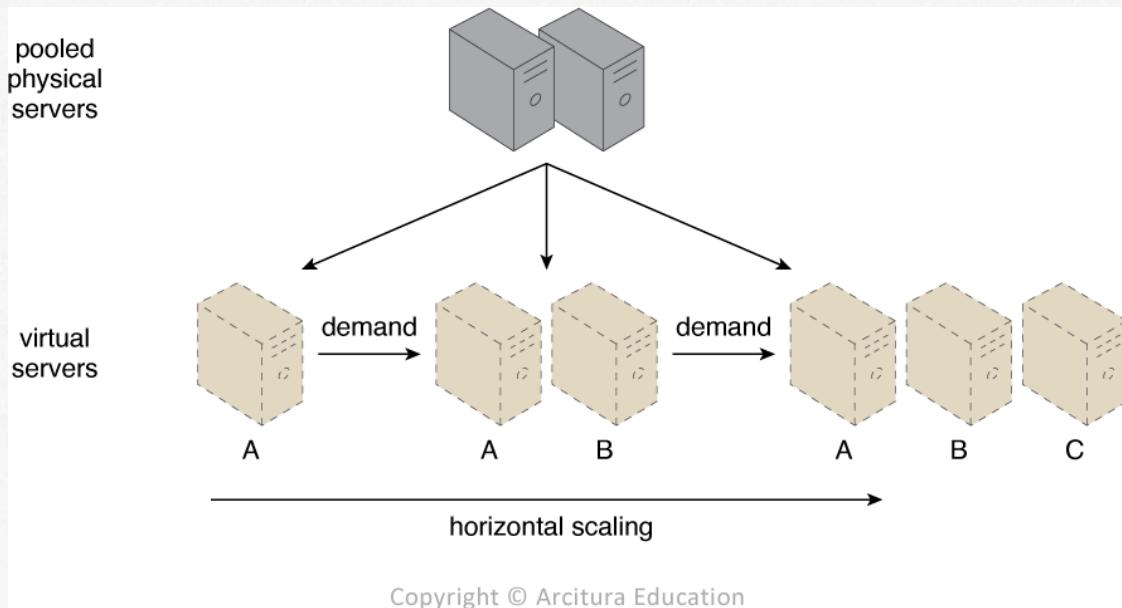
The following are types of scaling:

- Horizontal Scaling : scaling out and scaling in
- Vertical Scaling : scaling up and scaling down

Basic Concepts & Terminology

□ Horizontal Scaling :

- ✓ The allocating or releasing of IT resources that are of the same type is referred to as **horizontal scaling**, as shown in fig.
- ✓ The horizontal allocation of resources is referred to as **scaling out** and the horizontal releasing of resources is referred to as **scaling in**.
- ✓ Horizontal scaling is a common form of scaling within cloud environment.

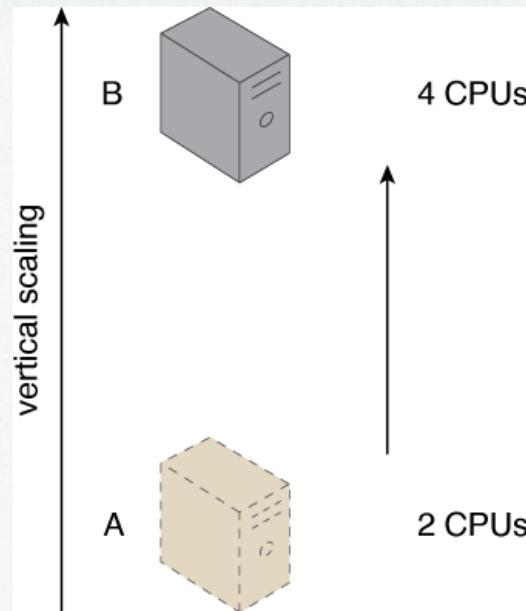


Basic Concepts & Terminology

□ Vertical Scaling :

- ✓ When an existing IT resources is replaced by another with higher or lower capacity, vertical scaling is considered to have occurred, as shown in the fig.
- ✓ The replacing of an IT resource with higher capacity is referred to as scaling up and the replacing an IT resource with lower capacity is referred to as scaling down.

Figure: An IT resource (a virtual server with two CPUs) is scaled up by replacing it with a more powerful IT resource with increased capacity for data storage (a physical server with four CPUs)

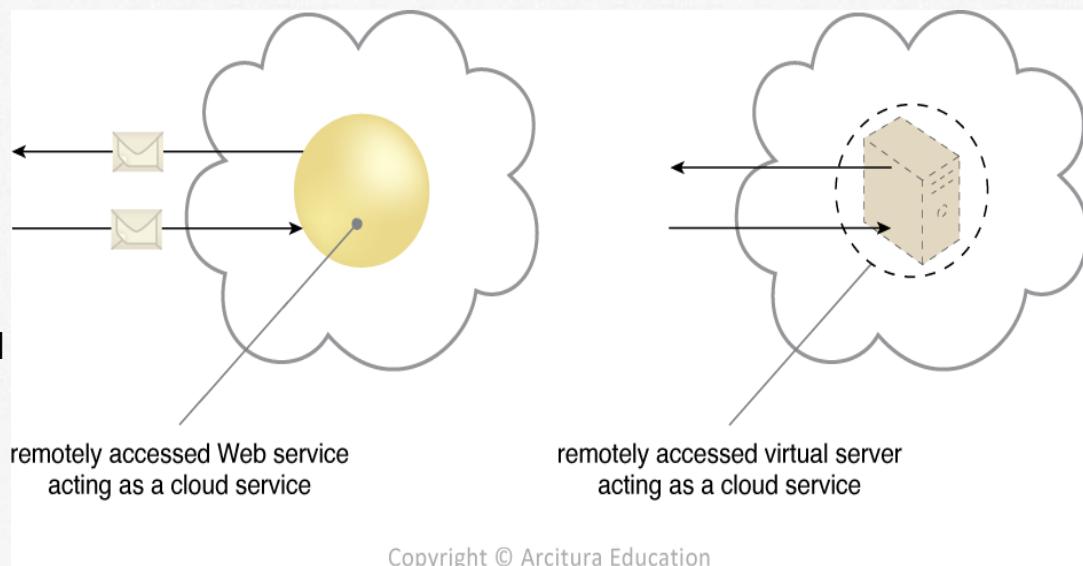


Basic Concepts & Terminology

□ Cloud Service:

- A **cloud service** is any IT resource that is made remotely accessible via a cloud.
- The term “service” within the context of cloud computing is especially broad.
- A cloud service can exist as a simple web-based software program with a technical interface invoked via the use of a messaging protocol, or as a remote access point for administrative tools or large environment and other IT resources.

Figure: A cloud service with a published technical interface is being accessed by a consumer outside of the cloud (left). A cloud service that exists as a virtual server is also being accessed from outside of the cloud's boundary (right)



Copyright © Arcitura Education

Basic Concepts & Terminology

□ Cloud Service consumers:

- A **cloud service consumer** is a temporary runtime role assumed by a software program when it accesses a cloud service.
- As shown in the fig. common type of cloud service consumers can include software programs and services capable of remotely accessing cloud services with published service contracts.



Figure: Examples of cloud service consumers. Depending on the nature of a given diagram, an artifact labeled as a cloud service consumer may be a software program or a hardware device (in which case it is running a software program capable of acting as a cloud service consumer).

Goals & Benefits

- Reduced Investments and proportional costs
- Increased scalability
- Increased availability and reliability

Cloud Reference Model

- The design of the NIST cloud computing reference architecture serves the following objectives:
 - to illustrate and understand the various cloud services in the context of an overall cloud computing conceptual model;
 - to provide a technical reference to USG agencies and other consumers to understand, discuss, categorize and compare cloud services;
 - and to facilitate the analysis of candidate standards for security, interoperability, and portability and reference implementations.

Cloud Reference Model

3. Cloud Computing Reference Architecture

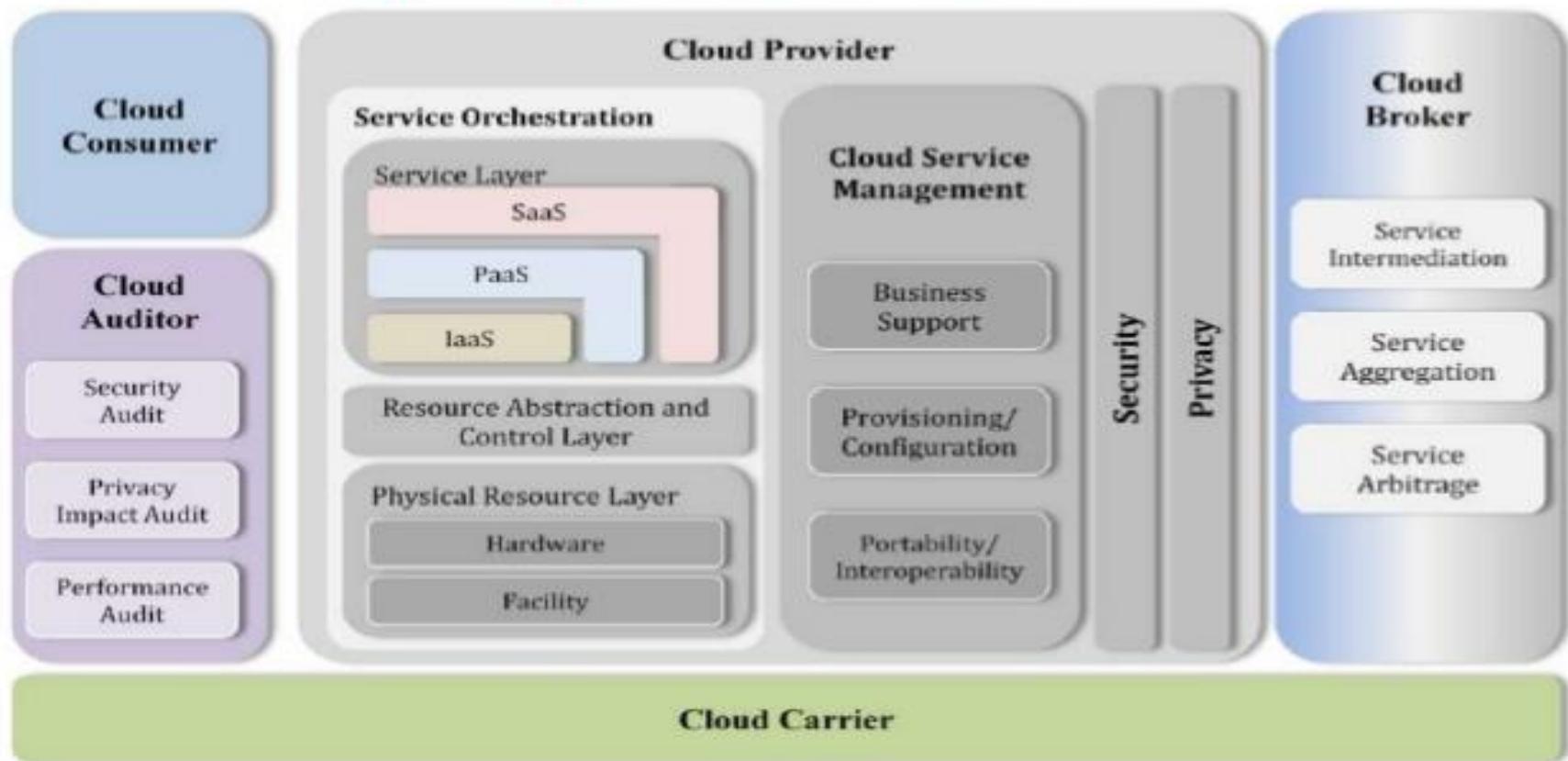


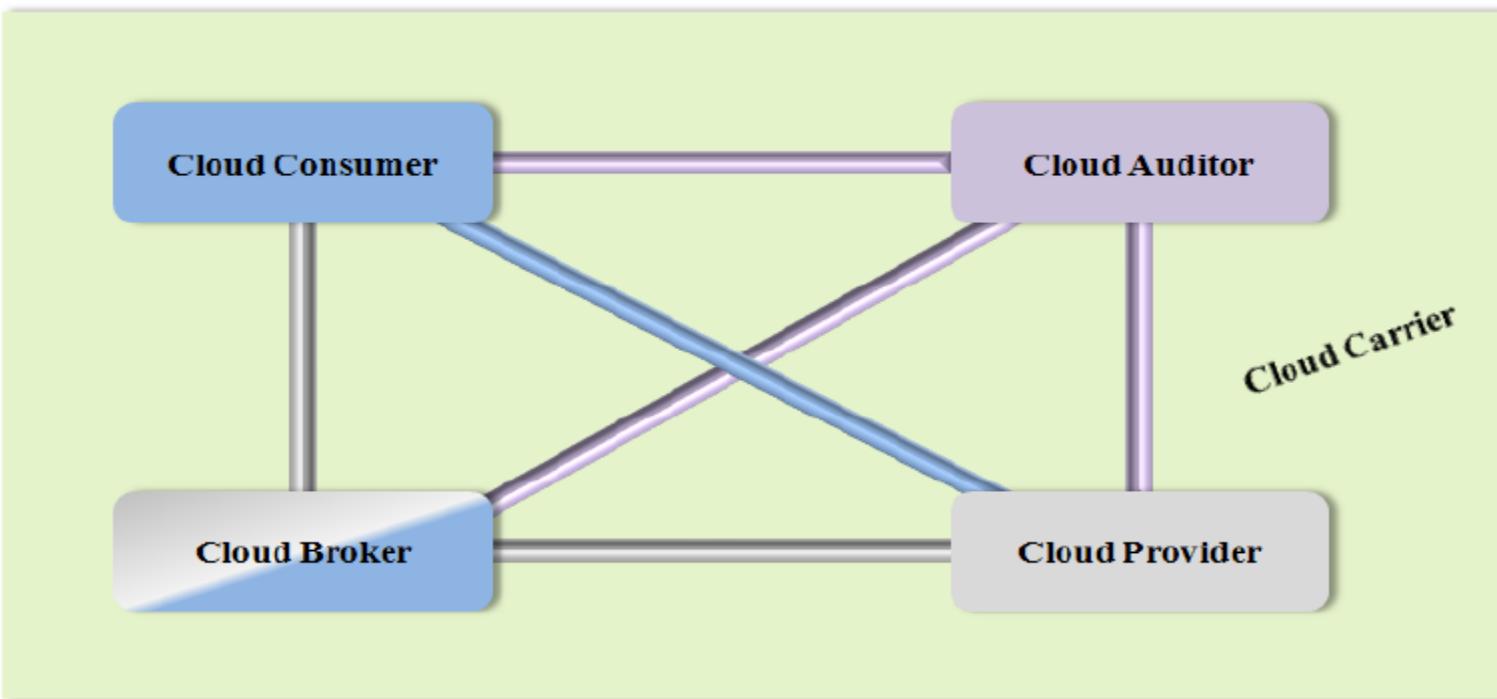
Figure 1: The Conceptual Reference Model

Roles in Cloud Computing

Actor	Definition
Cloud Consumer	A person or organization that maintains a business relationship with, and uses service from, <i>Cloud Providers</i> .
Cloud Provider	A person, organization, or entity responsible for making a service available to interested parties.
Cloud Auditor	A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation.
Cloud Broker	An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between <i>Cloud Providers</i> and <i>Cloud Consumers</i> .
Cloud Carrier	An intermediary that provides connectivity and transport of cloud services from <i>Cloud Providers</i> to <i>Cloud Consumers</i> .

Table 1: Actors in Cloud Computing

Communication between actors



- The communication path between a cloud provider and a cloud consumer
- The communication paths for a cloud auditor to collect auditing information
- The communication paths for a cloud broker to provide service to a cloud consumer

Figure 2: Interactions between the Actors in Cloud Computing

Cloud Usage Scenario Examples

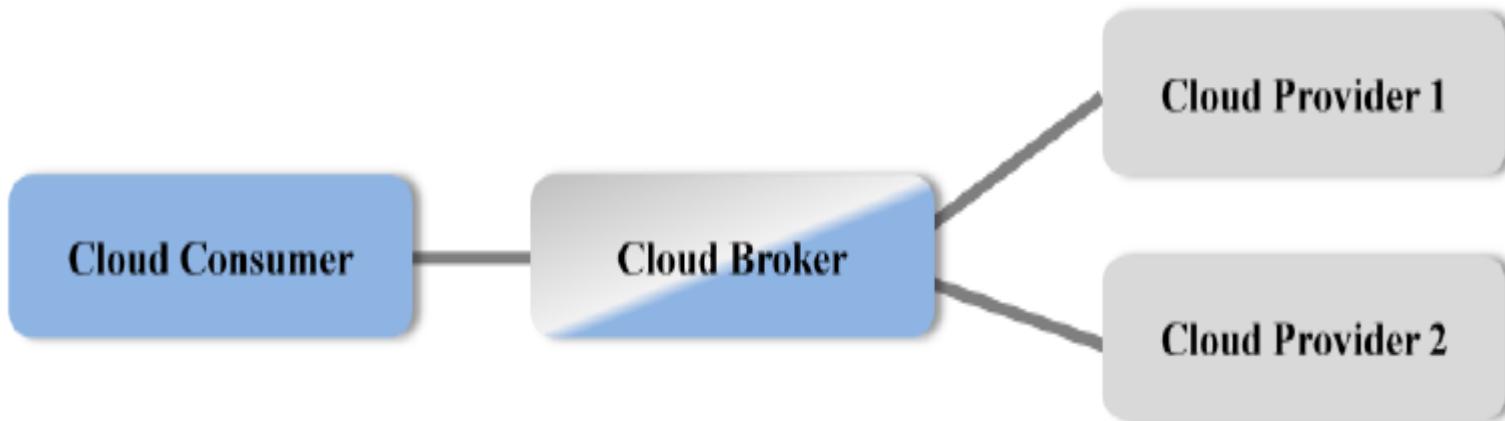
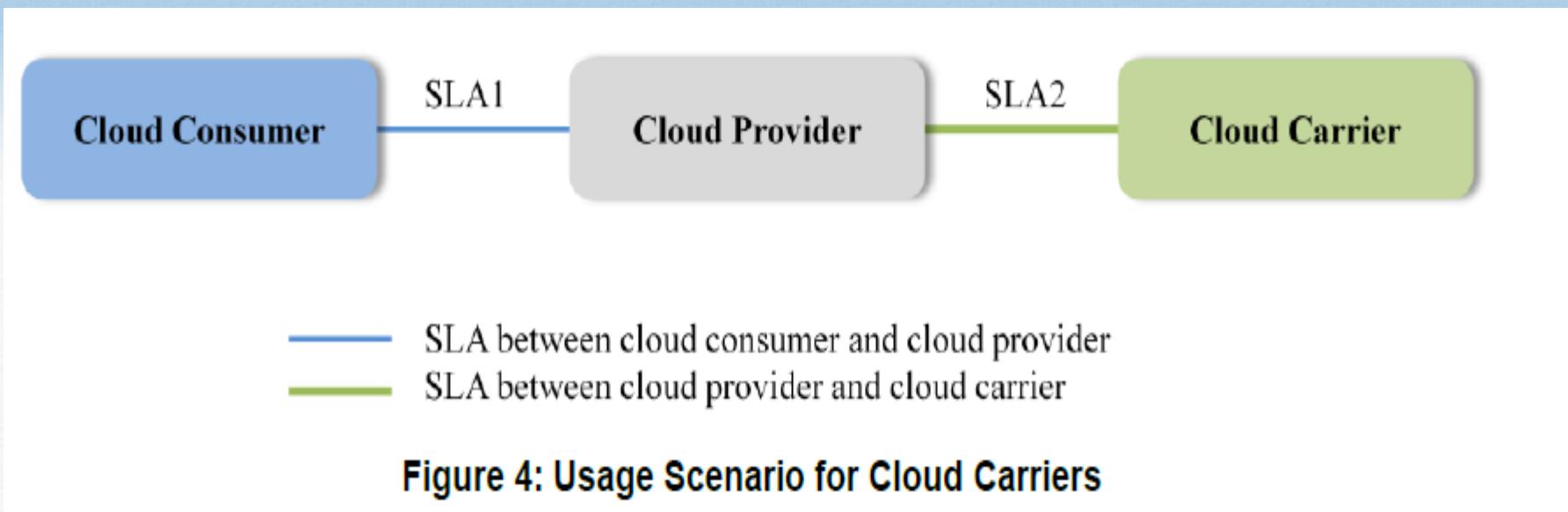


Figure 3: Usage Scenario for Cloud Brokers

Cloud Usage Scenario Examples



Cloud Usage Scenario Examples

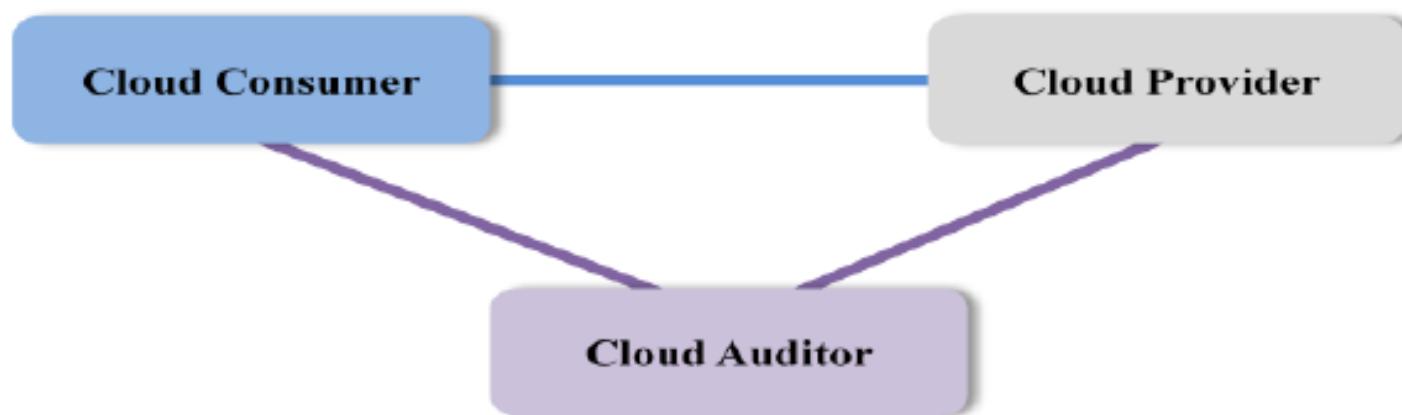
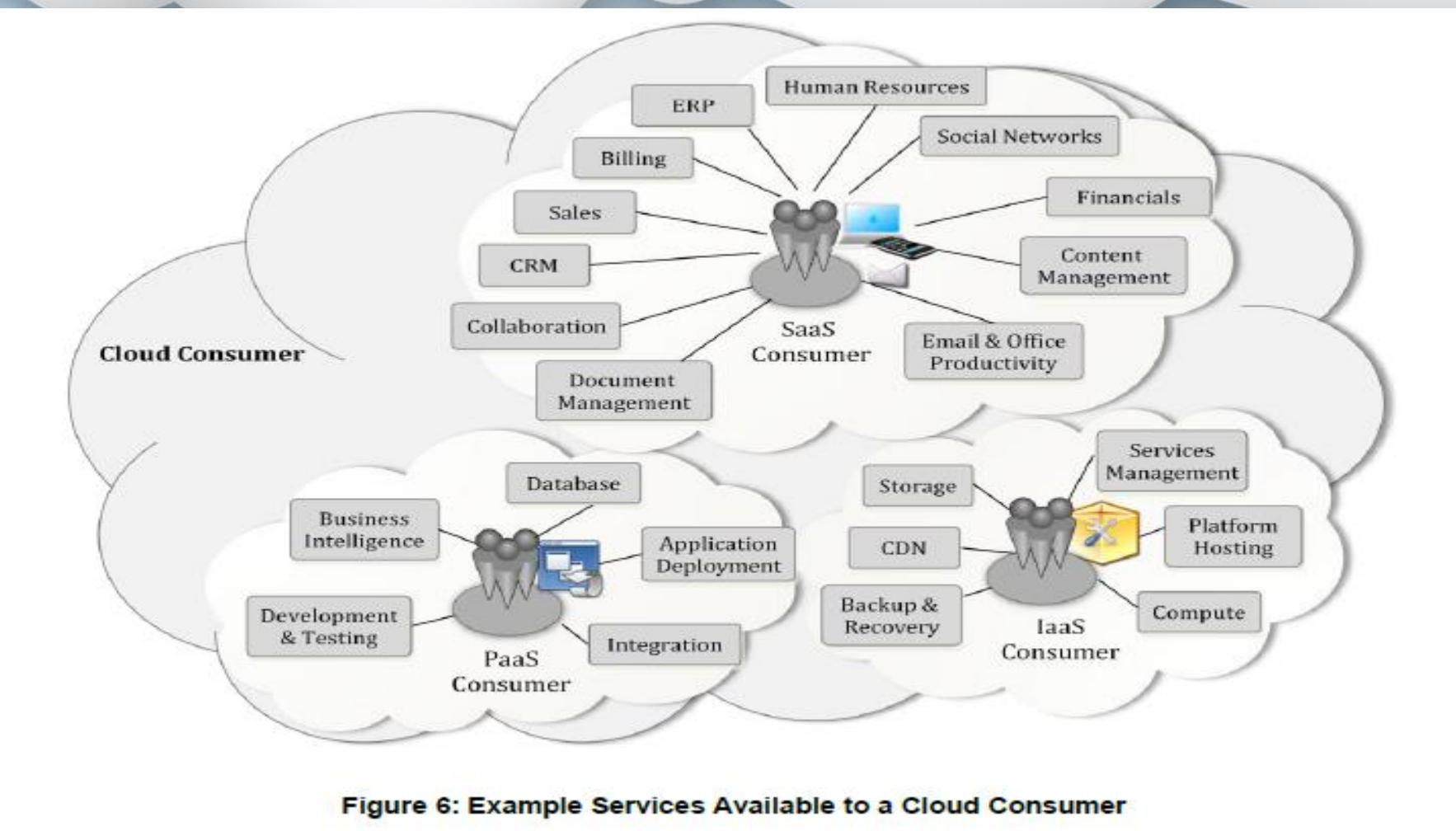


Figure 5: Usage Scenario for Cloud Auditors

Cloud Consumer



Cloud Service Provider

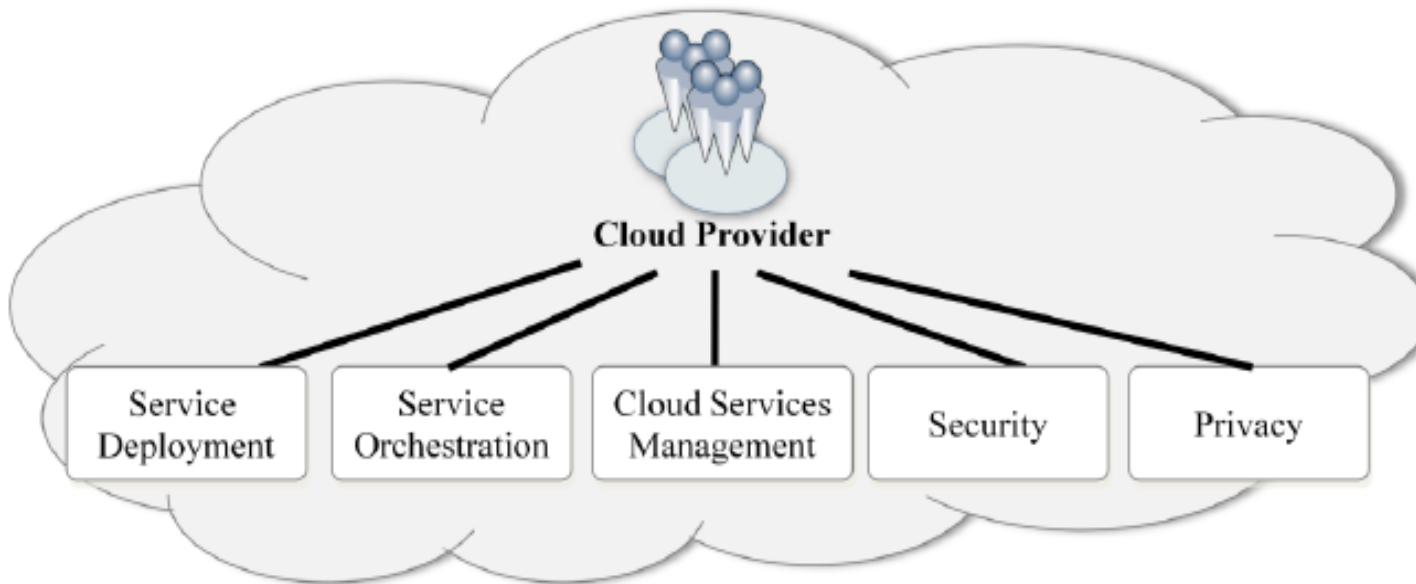


Figure 7: Cloud Provider - Major Activities

Cloud Auditor

- A cloud auditor is a party that can perform an independent examination of cloud service controls with the intent to express an opinion thereon.
- Audits are performed to verify conformance to standards through review of objective evidence.
- A cloud auditor can evaluate the services provided by a cloud provider in terms of security controls, privacy impact, performance, etc.
- **COMPANY EXAMPLES:**
 - ISACA (Infor. System audit & Control Association)
 - Product of ISACA is COBIT 5. (Control Objectives for Information and Related Technologies)
 - CloudChecker.com
 - Etc....

Cloud Broker

- As cloud computing evolves, the integration of cloud services can be too complex for cloud consumers to manage.
- A cloud consumer may request cloud services from a cloud broker, instead of contacting a cloud provider directly.
- A cloud broker is an entity that manages the use, performance and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers.
- Cloud Broker Companies: (*Total market is of \$2.03 billions in 2018*)
 - Appirio
 - AWS Marketplace
 - Bluewolf
 - CloudCompare
 - RED HAT Cloudforms is multicloud management platform.

Cloud Broker

In general, a cloud broker can provide services in three categories:

- **Service Intermediation:** A cloud broker enhances a given service by improving some specific capability and providing value-added services to cloud consumers. The improvement can be managing access to cloud services, identity management, performance reporting, enhanced security, etc.
- **Service Aggregation:** A cloud broker combines and integrates multiple services into one or more new services. The broker provides data integration and ensures the secure data movement between the cloud consumer and multiple cloud providers.
- **Service Arbitrage:** Service arbitrage is similar to service aggregation except that the services being aggregated are not fixed. Service arbitrage means a broker has the flexibility to choose services from multiple agencies. The cloud broker, for example, can use a credit-scoring service to measure and select an agency with the best score.

Cloud Carrier

- A cloud carrier acts as an intermediary that provides connectivity and transport of cloud services between cloud consumers and cloud providers.
- Cloud carriers provide access to consumers through network, telecommunication and other access devices. For example, cloud consumers can obtain cloud services through network access devices, such as computers, laptops, mobile phones, mobile Internet devices (MIDs), etc.
- The distribution of cloud services is normally provided by network and telecommunication carriers or a *transport agent*, where a transport agent refers to a business organization that provides physical transport of storage media such as high-capacity hard drives.

Scope of Control between Provider & Consumer

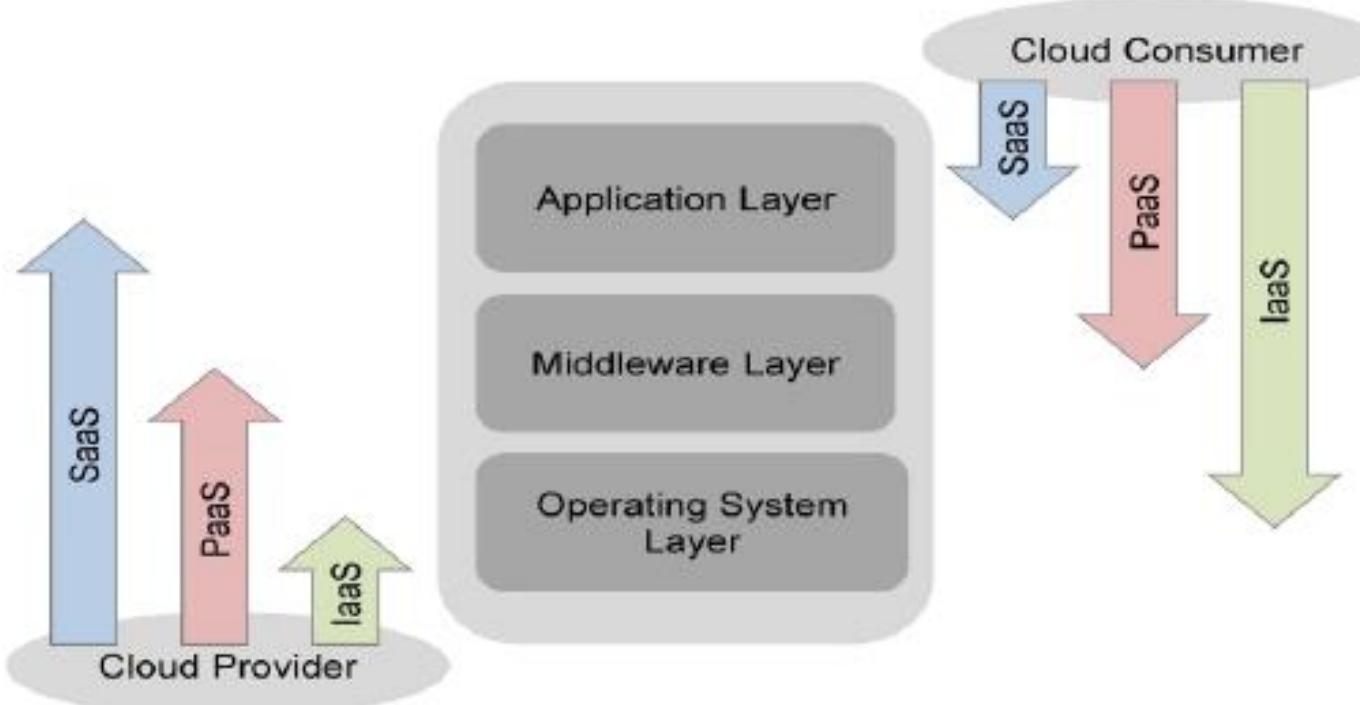
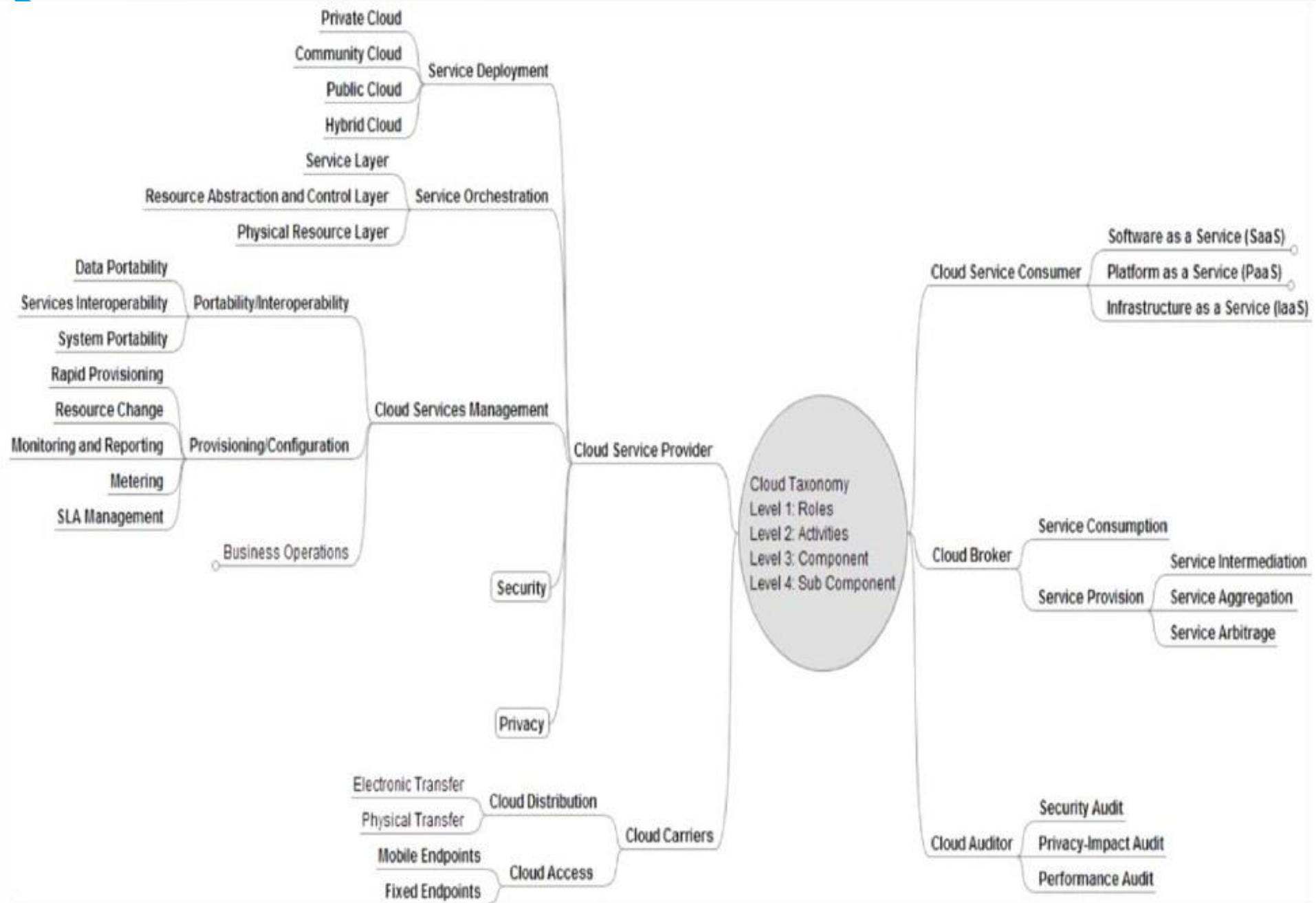


Figure 8: Scope of Controls between Provider and Consumer

Cloud Taxonomy

- Taxonomy is the science of categorization, or classification, of things based on a predefined system.
- Typically, taxonomy contains a controlled vocabulary with a hierarchical tree-like structure.



Cloud Taxonomy

- Figure presents the taxonomy associated with the cloud computing reference architecture discussed in this document. In the figure, a four-level taxonomy is presented to describe the key concepts about cloud computing.
- **Level 1:** *Role*, which indicates a set of obligations and behaviors as conceptualized by the associated actors in the context of cloud computing.
- **Level 2:** *Activity*, which entails the general behaviors or tasks associated to a specific role.
- **Level 3:** *Component*, which refer to the specific processes, actions, or tasks that must be performed to meet the objective of a specific activity.
- **Level 4:** *Sub-component*, which present a modular part of a component.

Roles

- **Cloud Consumer** - Person or organization that maintains a business relationship with, and uses service from, Cloud Service Providers.
-
- **Cloud Provider** – Person, organization or entity responsible for making a service available to service consumers.
-
- **Cloud Carrier** – The intermediary that provides connectivity and transport of cloud services between Cloud Providers and Cloud Consumers.
-
- **Cloud Broker** – An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers.
-
- **Cloud Auditor** – A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation.

Cloud Service provider activities

1. **Service Deployment** – All of the activities and organization needed to make a cloud service available
2. **Service Orchestration** - Refers to the arrangement, coordination and management of cloud infrastructure to provide different cloud services to meet IT and business requirements.
3. **Cloud Service Management** – Cloud Service Management includes all the service-related functions that are necessary for the management and operations of those services required by or proposed to customers.

Cloud Service provider activities

4. **Security** – Refers to information security. “information security” means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide:
 - (A) **integrity**, which means guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity;
 - (B) **confidentiality**, which means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information;
 - (C) **availability**, which means ensuring timely and reliable access to and use of information.
5. **Privacy** - Information privacy is the assured, proper, and consistent collection, processing, communication, use and disposition of disposition of personal information (PI) and personally-identifiable information (PII) throughout its life cycle.

Cloud Carrier activities:

1. **Cloud Distribution** – The process of transporting cloud data between Cloud Providers and Cloud Consumers.
2. **Cloud Access** – To make contact with or gain access to Cloud Services.

Risks & Challenges

- Increased security vulnerabilities.

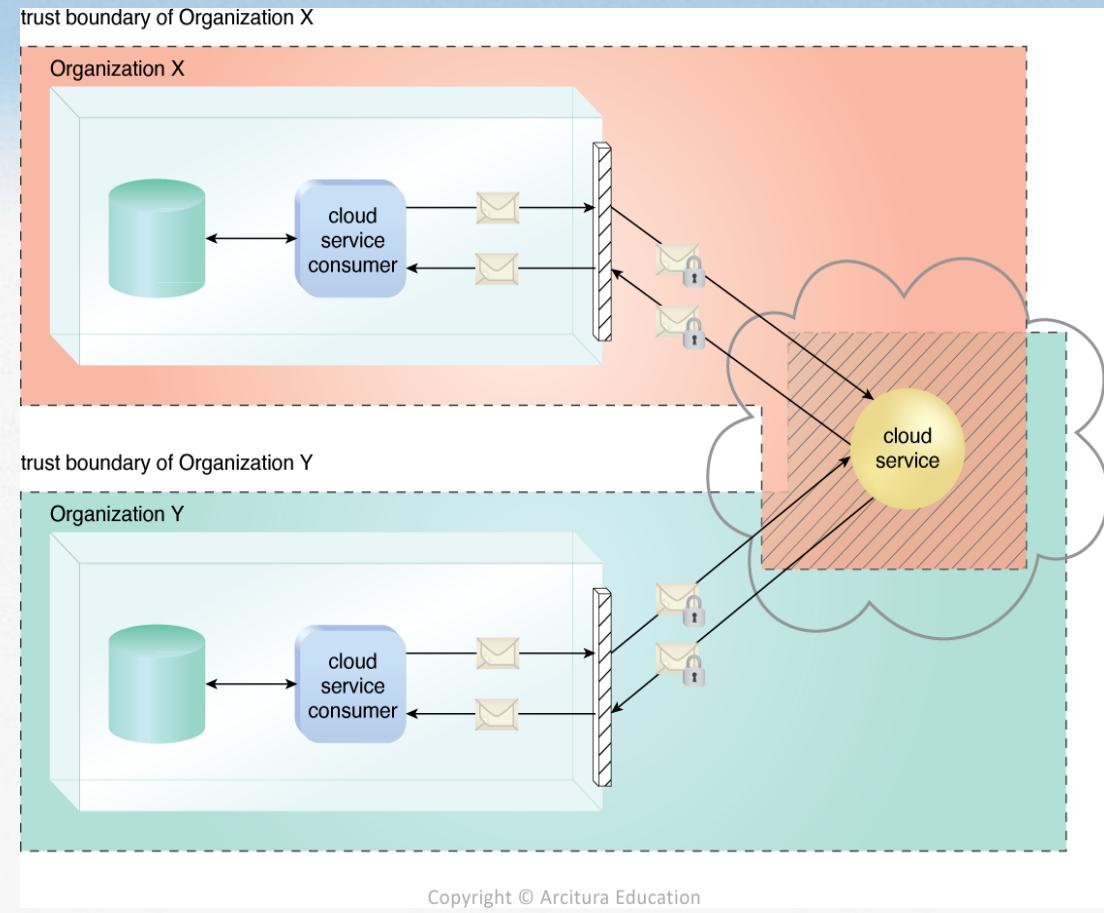
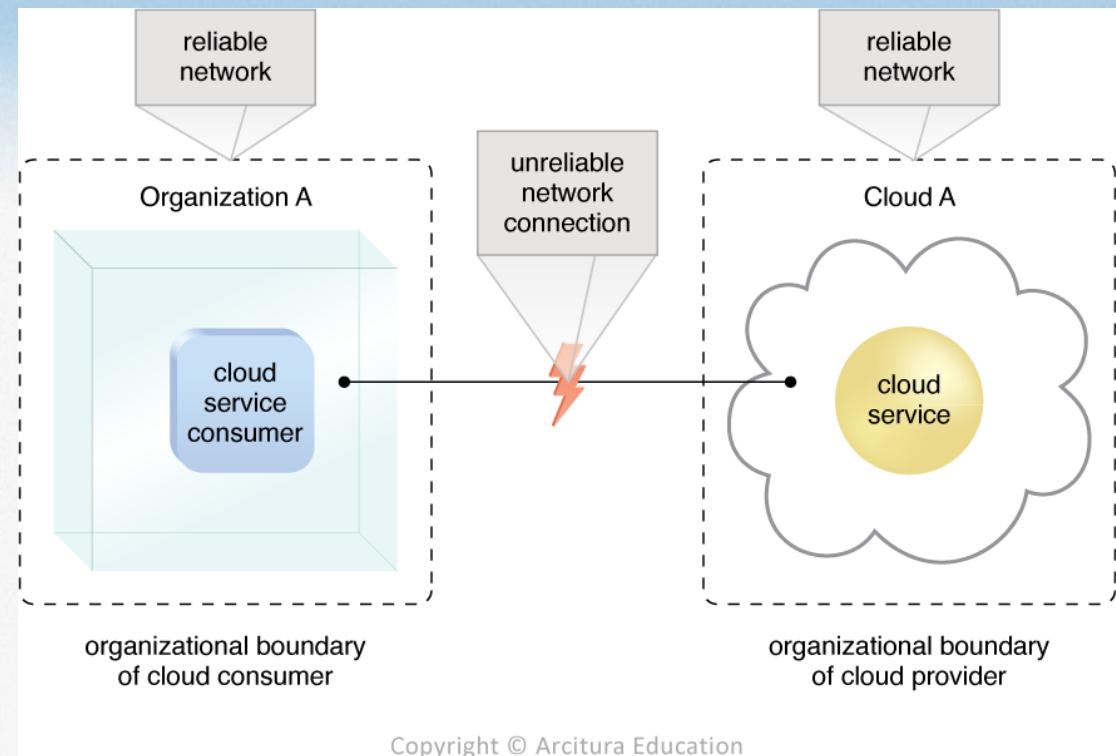


Figure: The shaded area with diagonal lines indicates the overlap of two organization's trust boundaries.

Risks & Challenges

- Reduced Operational Governance Control

Figure: An unreliable network connection compromises the quality of communication between cloud consumer and cloud provider environments.



Copyright © Arcitura Education

Risks & Challenges

- Limited Portability Between Cloud Providers

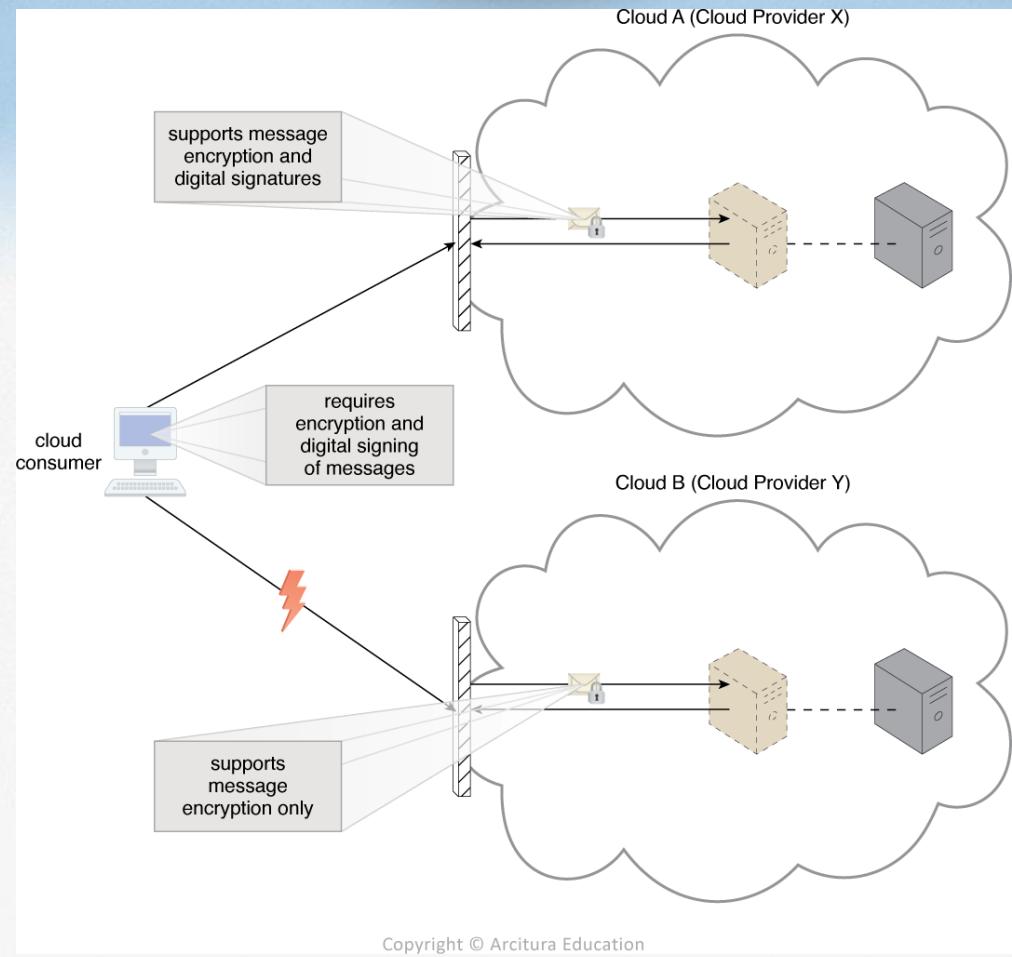


Figure: A cloud consumers application has a decreased level of portability when accessing a potential migration from Cloud A to Cloud B, because the cloud provider of Cloud B does not support the same security technologies as Cloud A.

Risks & Challenges

- Multiregional Compliance and Legal Issues

Summary of Key Points

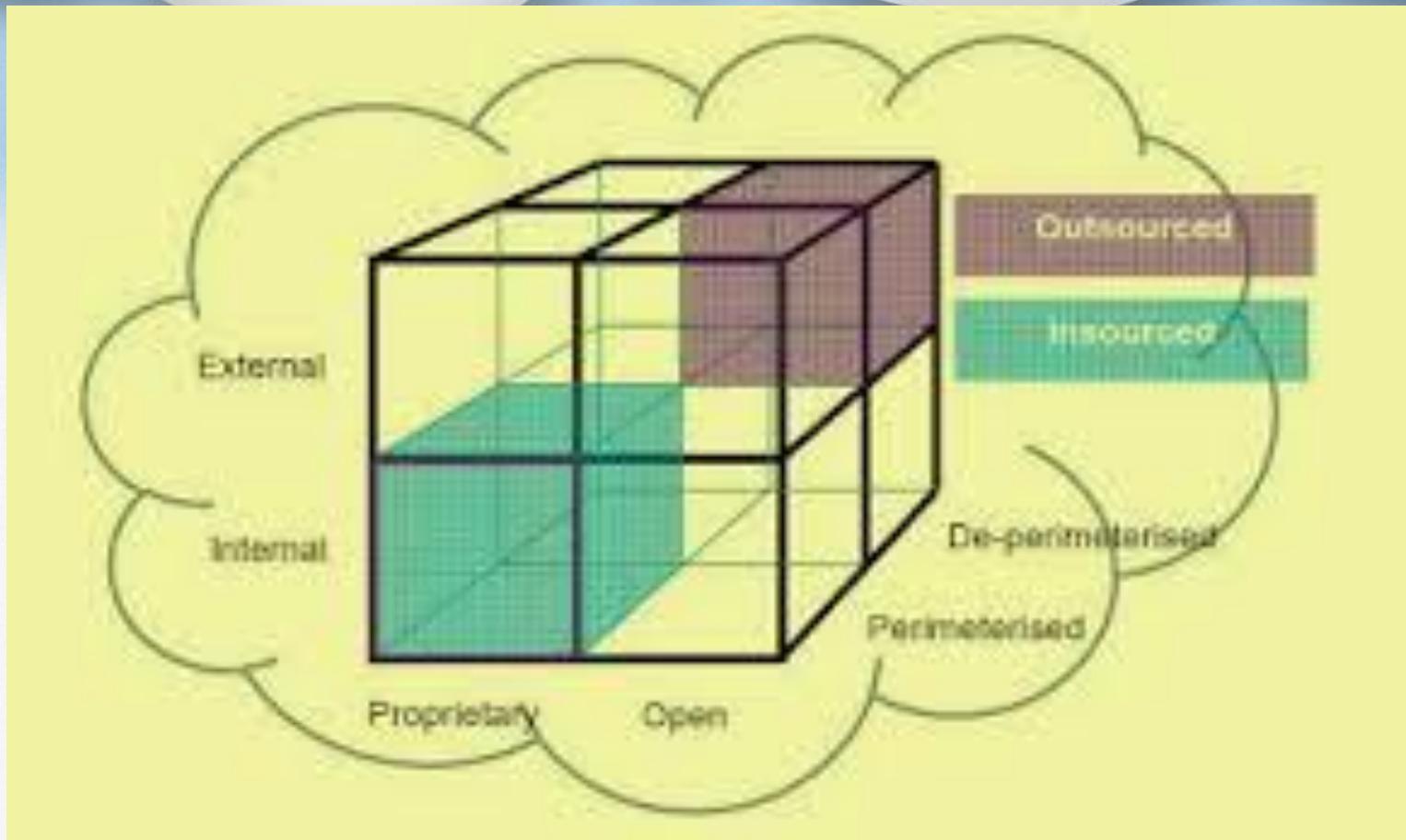
- Cloud environments can introduce distinct security challenges, some of which pertain to overlapping trust boundaries imposed by a cloud provider sharing IT resources with multiple cloud consumers.
- A cloud consumer's operational governance can be limited within cloud environments due to the control exercised by a cloud provider over its platforms.
- The portability of cloud-based IT resources can be inhibited by dependencies upon proprietary characteristics imposed by a cloud.
- The geographical location of data and IT resources can be out of cloud consumer's control when hosted by a third-party cloud provider. This can introduce various legal and regulatory compliance concern.

Muddiest Point

Cloud Characteristics

- The following six specific characteristic are common to the majority of cloud environments:
 - 1) On-demand usage
 - 2) Ubiquitous access
 - 3) Multitenancy (and resource pooling)
 - 4) Elasticity
 - 5) Measured usage
 - 6) Resiliency

Cloud Cube Model



Cloud Cube Model

- The Jericho Forum has designed the Cloud Cube Model to help select cloud formations for security cooperation.
- Their fascinating new cloud model helps IT managers and business tycoons assess the benefits of cloud computing.
- The Cloud Cube Model looks at the several different "cloud formations".
- They amount to the cloud service and deployment models.
- The sourcing dimension addresses the delivery of service.
- The Cloud Cube Model may be designed to let users show that the traditional notion of network ranges & its boundaries with network firewall no longer applies in Cloud computing.

Cloud Cube Model

- **Cloud Cube Model**, designed and developed by **Jericho forum**.
- Which helps to categorize the cloud network based on the four-dimensional factor: Internal/External, Proprietary/Open, De-Perimeterized/Perimeterized, and Insourced/Outsourced.

• **Dimension 1: Internal/External**

- This dimension defines the physical location of the data; where does the cloud form exist – inside or outside organization boundaries? If the cloud form is within the organization's physical boundaries, then it is internal.
- If it is outside the organization's physical boundaries, then it is external. It's important to note that the assumption that internal is necessarily more secure than external is false. The most secure usage model is the effective use of both internal and external cloud forms.

ii. Proprietary/Open

- The second type of cloud formation is **proprietary and open**. The proprietary or open dimension states about the state of ownership of the **cloud technology** and interfaces. It also tells the degree of interoperability, while enabling data transportability between the system and other cloud forms..
- The **proprietary dimension** means, that the organization providing the **service is securing** and protecting the data under their ownership.
- The **open dimension** is using a technology in which there are more suppliers. Moreover, the user is not constrained in being able to share the data and collaborate with selected partners using the open technology.ther cloud forms.

iii. De-Perimeterized / Perimeterized

- The third type of cloud formation is **De-perimeterized and Perimeterized**.
- To reach this form, the user needs collaboration oriented architecture and Jericho forum commandments.
The Perimeterised and De-perimeterized dimension tells us whether you are operating inside your traditional it mindset or outside it.
- **Perimeterized dimension** means, continuing to operate within the traditional it boundary, orphan signaled by network firewalls. With the help of VPN and operation of the virtual server in your own IP domain, the user can extend the organizations perimeter into external Cloud Computing domain. This means that the user is making use of the own services to control access.

iii. De-Perimeterized / Perimeterized

- **De-perimeterized dimension** means the system perimeter is architected on the principles outlined in the Jericho forums commandments. In De-perimeterized dimension, the data will be encapsulated with metadata and mechanisms, which will further help to protect the data and limit the inappropriate usage.

iv. Insourced/Outsourced

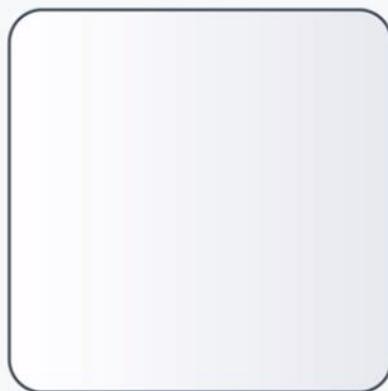
- The **Insourced and outsourced dimensions** have two states in each of the eight cloud forms. In the *outsourced dimension* the services provided by the third party, whereas in the *insourced dimension* the services provided by the own staff under the control.
- In this few organizations that are traditional bandwidth software or hardware, providers will run fluently on becoming cloud service providers.
- The organizations which are seeking to procedure cloud services must have the ability to set legally binding collaboration agreement. In this, an organization should ensure that data is deleted from the service provider's Infrastructure.

Questions For Cloud Cube Model

- The Jericho forum states that there are three key questions, which a customer should ask their **Cloud Computing** supplier. So, that they must be aware that the data is secure and protected. The three questions are-
- Q 1. Wherein the cloud cube model is the cloud supplier operating while providing the services?
- Q 2. How will the clouds suppliers get a surety when the customer is using services in a cloud from that has maintained the features as per the expectations?
- Q 3. How can a customer ensure that the data which is stored in the cloud services will be available at the time of mishappenings such as bankruptcy or change in business direction?

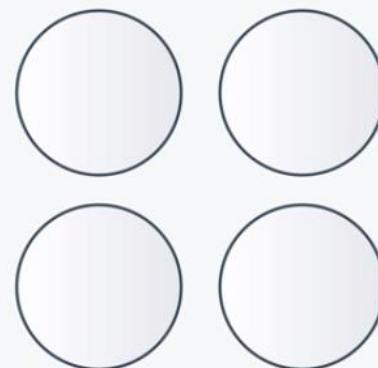
MONOLITHIC, SOA & MICROSERVICES

Monolithic vs. SOA vs. Microservices



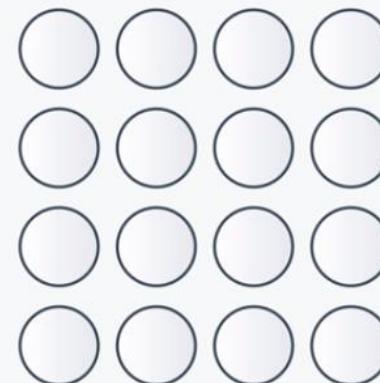
Monolithic

Single Unit



SOA

Coarse-grained



Microservices

Fine-grained

Activate Windows
Go to Settings to activate Windows.



SOA Vs Microservice



SOA is like an orchestra where each artist is performing with his/her instrument while the music director guides them all.

With Microservices each dancer is independent and know what they need to do. If they miss some steps they know how to get back on the sequence.

Activate Windows
Go to Settings to activate Windows.

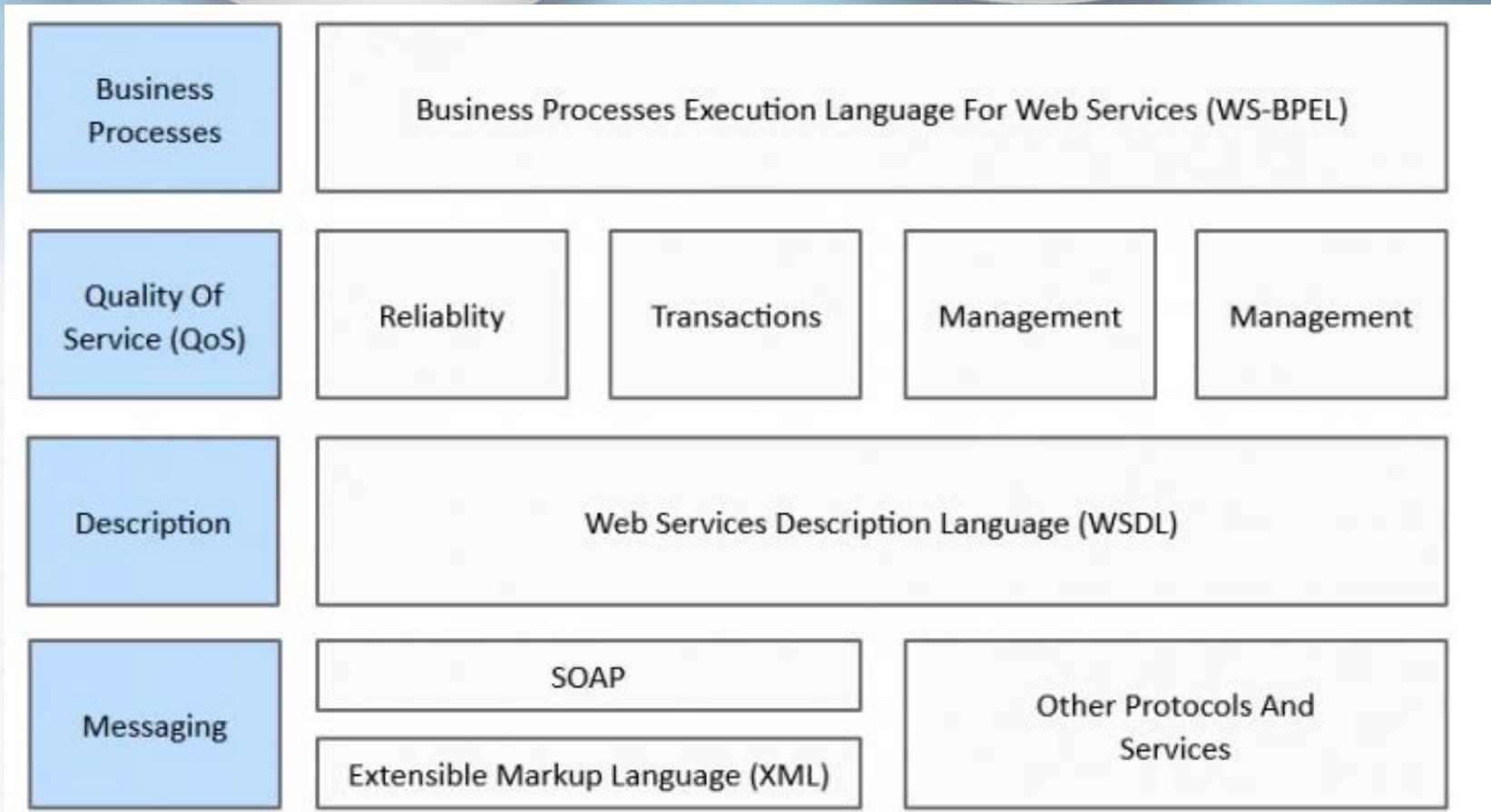
Service Oriented Architecture

- **Service-oriented architecture (SOA)** is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.
- A SOA service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online. SOA is also intended to be independent of vendors, products and technologies.

Service Oriented Architecture

- A service has four properties according to one of many definitions of SOA:
 - It logically represents a business activity with a specified outcome.
 - It is self-contained.
 - It is a black box for its consumers, meaning the consumer does not have to be aware of the service's inner workings.
 - It may consist of other underlying services.
- Different services can be used in conjunction to provide the functionality of a large software application, a principle SOA shares with modular programming.

SOA Architecture and Protocols



Service Oriented Architecture

- SOA architecture is viewed as five horizontal layers. These are described below:
 - 1) **Consumer Interface Layer:** These are GUI based apps for end users accessing the applications.
 - 2) **Business Process Layer:** These are business-use cases in terms of application.
 - 3) **Services Layer:** These are whole-enterprise, in service inventory.
 - 4) **Service Component Layer:** are used to build the services, such as functional and technical libraries.
 - 5) **Operational Systems Layer:** It contains the data model.

Service Oriented Architecture

- A manifesto was published for service-oriented architecture in October, 2009. This came up with six core values which are listed as follows:
 - 1) **Business value** is given more importance than technical strategy.
 - 2) **Strategic goals** are given more importance than project-specific benefits.
 - 3) **Intrinsic interoperability** is given more importance than custom integration.
 - 4) **Shared services** are given more importance than specific-purpose implementations.
 - 5) **Flexibility** is given more importance than optimization.
 - 6) **Evolutionary refinement** is given more importance than pursuit of initial perfection.

Guiding Principles of SOA:

- **Standardized service contract:** Specified through one or more service description documents.
- **Loose coupling:** Services are designed as self-contained components, maintain relationships that minimize dependencies on other services.
- **Abstraction:** A service is completely defined by service contracts and description documents. They hide their logic, which is encapsulated within their implementation.
- **Reusability:** Designed as components, services can be reused more effectively, thus reducing development time and the associated costs.

Guiding Principles of SOA:

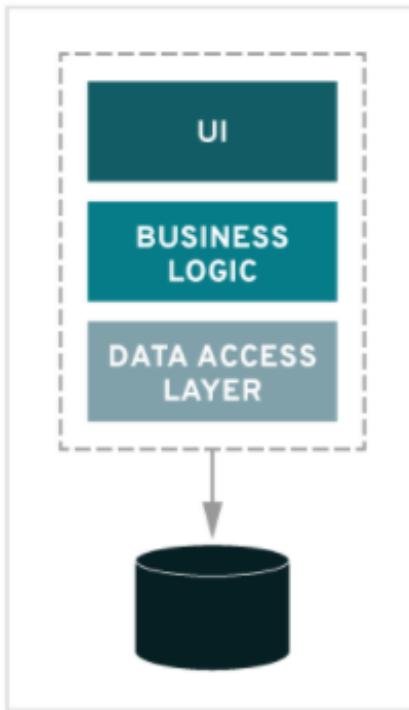
- **Autonomy:** Services have control over the logic they encapsulate and, from a service consumer point of view, there is no need to know about their implementation.
- **Discoverability:** Services are defined by description documents that constitute supplemental metadata through which they can be effectively discovered. Service discovery provides an effective means for utilizing third-party resources.
- **Composability:** Using services as building blocks, sophisticated and complex operations can be implemented. Service orchestration and choreography provide a solid support for composing services and achieving business goals.

Microservices

- **Microservice** architecture – a variant of the service-oriented architecture (SOA) structural style – arranges an application as a collection of loosely coupled services.
- In a microservices architecture, services are fine-grained and the protocols are lightweight.

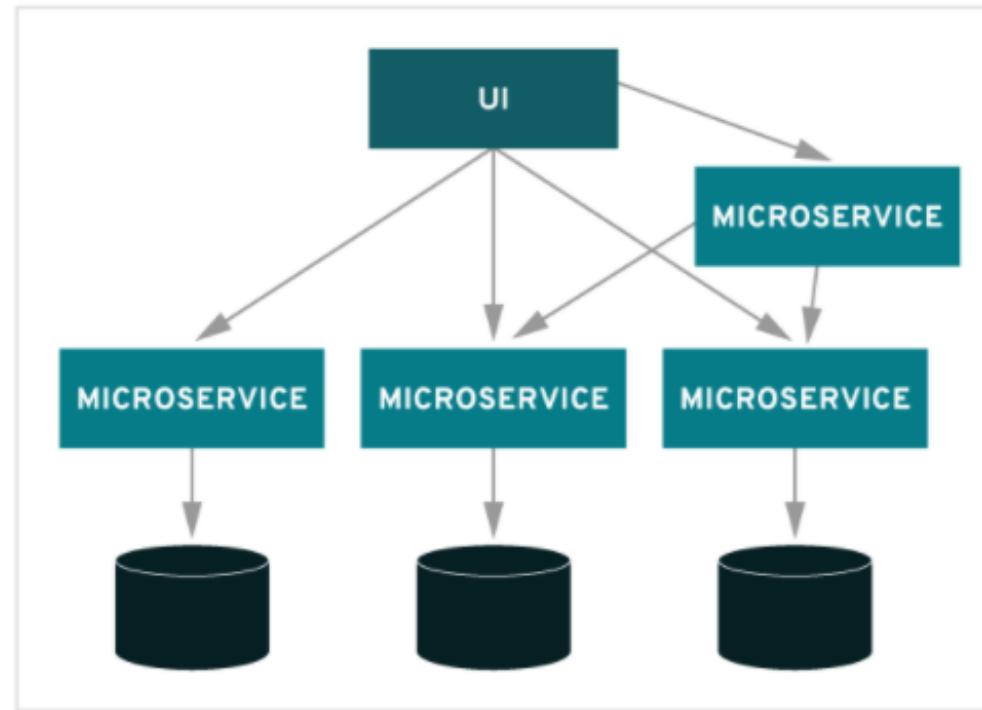
Microservices

MONOLITHIC



MICROSERVICES

VS.



Microservices Features



WEB 1.0 To 5.0

- **Web 0.0** – Developing the internet
- **Web 1.0** – The shopping carts & static web
- **Web 2.0** – The writing and participating web
- **Web 3.0** – The semantic executing web
- **Web 4.0** – “Mobile Web”
- **Web 5.0**- Open, Linked and Intelligent
Web = Emotional Web. “The next web”

THANK YOU!!!!

Unit-III Terraform

Cloud Computing and DevOps

Ms. Vidya S. Gaikwad

vidya.gaikwad@viit.ac.in

Department of Computer Engineering



BRAC'T's, Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)

Ms. Vidya S. Gaikwad, Department of Computer Engineering, VIIT , Pune-48

UNIT-III

Terraform

- **Introduction:**
 - What You Need to Know About Infrastructure as Code
 - Infrastructure as Code Defined, Declarative vs. Imperative
 - Idempotence and Consistency
 - Benefits of Infrastructure as Code.
- **Understand Terraform's purpose (vs other IaC) :**
 - Explain multi-cloud and provider-agnostic benefits
- **Understand Terraform basics:**
 - Handle Terraform and provider installation and versioning
 - Describe how Terraform finds and fetches providers, provisioners
 - when to use local-exec or remote-exec.
- **Deploying Your First Terraform Configuration:**
 - Installing Terraform
 - Using the CLI
 - Terraform Object Types
 - General Block Syntax
 - Terraform Workflow
 - Deploying the Base Configuration
 - Validating the Deployment, Implement and maintain state, Using Input Variables and Outputs, Module.

- Infrastructure as Code (IaC) is a widespread terminology among DevOps professionals.
- It is the process of managing and provisioning the complete IT infrastructure (comprises both physical and virtual machines) using machine-readable definition files.
- It is a software engineering approach toward operations.
- It helps in automating the complete data centre by using programming scripts.
- With all the features that Infrastructure as Code provides, it has multiple challenges:
 - Need to learn to code
 - Don't know the change impact.
 - Need to revert the change
 - Can't track changes
 - Can't automate a resource
 - Multiple environments for infrastructure

What is Terraform?

- Terraform has been created to solve these challenges.
- Terraform is an **open-source infrastructure as Code tool developed by HashiCorp**.
- It is used to define and provision the complete infrastructure using an easy-to-learn declarative language.
- It is an infrastructure provisioning tool where you can store your cloud infrastructure setup as codes.
- It's very similar to tools such as **CloudFormation**, which you would use to automate your AWS infrastructure, but you can only use that on AWS.
- With Terraform, you can use it on other cloud platforms as well.

- **Below are some of the benefits of using Terraform.**
- Does orchestration, not just configuration management.
- Supports multiple providers such as AWS, Azure, GCP, DigitalOcean and many more
- Provide immutable infrastructure where configuration changes smoothly
- Uses easy to understand language, HCL (HashiCorp configuration language)
- Easily portable to any other provider
- Supports Client only architecture, so no need for additional configuration management on a server

- Below are the core concepts/terminologies used in Terraform:
- **Variables:** Also used as input-variables, it is key-value pair used by Terraform modules to allow customization.
- **Provider:** It is a plugin to interact with APIs of service and access its related resources.
- **Module:** It is a folder with Terraform templates where all the configurations are defined
- **State:** It consists of cached information about the infrastructure managed by Terraform and the related configurations.

- **Resources:** It refers to a block of one or more infrastructure objects (compute instances, virtual networks, etc.), which are used in configuring and managing the infrastructure.
- **Data Source:** It is implemented by providers to return information on external objects to terraform.
- **Output Values:** These are return values of a terraform module that can be used by other configurations.
- **Plan:** It is one of the stages where it determines what needs to be created, updated, or destroyed to move from real/current state of the infrastructure to the desired state.
- **Apply:** It is one of the stages where it applies the changes real/current state of the infrastructure in order to move to the desired state.



Terraform Architecture:[2]

The picture can't be displayed.

- Terraform is a **plugin-based tool**.
- So, it has a **core application, terraform, and hundreds of plugins**.
- The core application provides a unified layer to manage the IaC codes, and it's responsible for installing required plugins, invoking them, managing state, etc.
- On the other hand, **plugins are here to communicate with real infrastructure platforms and applications like AWS, GCP, Azure, Grafana, Jenkins, Gitlab, etc.**
- Some of these **plugins are maintained officially by the HashiCorp team, and some of them are maintained by third parties**.
- **Everyone can write and distribute his own plugin.**
- You can find **hundreds of plugins on the Terraform Registry website**.

•

- Terraform plugins are separated into two types: **Providers** and **Provisioners**.
- The Provider is a plugin which is responsible for connecting to the real infrastructure or application through the API and creating, modifying, and deleting objects and resources,
- The Provisioner is a plugin which is responsible for connecting to a provisioned infrastructure and making some changes to it.
- For example, you can use the AWS PROVIDER to provision an EC2 instance and use remote-exec PROVISIONER to execute some commands on it after provisioning. You should note that it's not the best practice to use PROVISIONERS at all, and they are provided as a last resort solution.



Terraform Lifecycle [1]

[The picture can't be displayed.

- **Terraform init** initializes the working directory which consists of all the configuration files.
- **Terraform plan** is used to create an execution plan to reach a desired state of the infrastructure.
- Changes in the configuration files are done in order to achieve the desired state.
- **Terraform apply** then makes the changes in the infrastructure as defined in the plan, and the infrastructure comes to the desired state.
- **Terraform destroy** is used to delete all the old infrastructure resources, which are marked tainted after the apply phase.



How Terraform Works?[1]

The picture can't be displayed.

How Terraform Works?[1]

- Terraform has two main components that make up its architecture:
 - **Terraform Core**
 - **Providers**
- **Terraform Core**
- Terraform core uses two input sources to do its job.
- The **first** input source is a Terraform configuration that you, as a user, configure.
- Here, you define what needs to be created or provisioned.
- And the **second** input source is a state where terraform keeps the up-to-date state of how the current set up of the infrastructure looks like

How Terraform Works?[1]

- Terraform core takes the input, and it figures out the plan of what needs to be done.
- It compares the state, what is the current state, and what is the configuration that you desire in the end result.
- It figures out what needs to be done to get to that desired state in the configuration file.
- It figures what needs to be created, what needs to be updated, what needs to be deleted to create and provision the infrastructure.

- **Providers**
- The second component of the architecture are providers for specific technologies.
- This could be cloud providers like AWS, Azure, GCP, or other infrastructure as a service platform.
- It is also a provider for more high-level components like **Kubernetes** or other **platform-as-a-service tools**, even some software as a self-service tool.
- It gives you the possibility to create infrastructure on different levels.
- For example – create an AWS infrastructure, then deploy Kubernetes on top of it and then create services/components inside that Kubernetes cluster.

References

- 1) <https://geekflare.com/terraform-for-beginners/>
- 2) <https://itnext.io/terraform-tutorial-part-1-intro-and-basic-concepts-7a27ae7722b6>



ARTURO PIE @arturo_pie

IAC WITH TERRAFORM



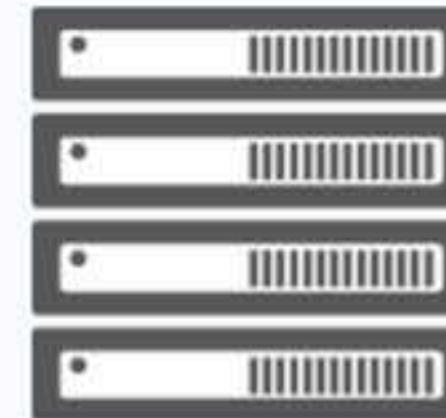
- ▶ Based in Toronto, Canada
- ▶ Goal: Build software that dramatically reduces waste in the global supply chain
- ▶ We serve Kellogg's, P&G, DHL, L'Oreal and others
- ▶ Software Craftsmanship
- ▶ Learning organization
- ▶ Modern Agile methodology
- ▶ TDD and pair programming

INFRASTRUCTURE

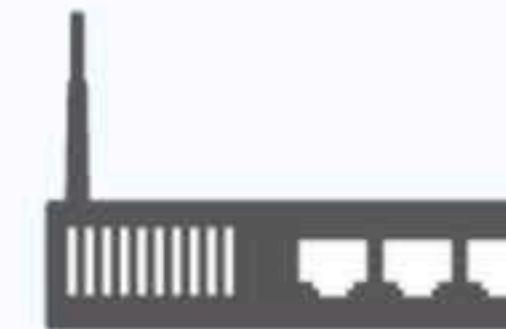
Orchestration & Automation



Storage/
Availability



Servers



Networking



Security



Management/
Monitoring

**App Resource
Management**

INFRASTRUCTURE AS CODE

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
}

data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-trusty-14.04-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "web" {
  ami          = "${data.aws_ami.ubuntu.id}"
  instance_type = "t2.large"
  security_groups = ["${aws_security_group.allow_all.id}"]

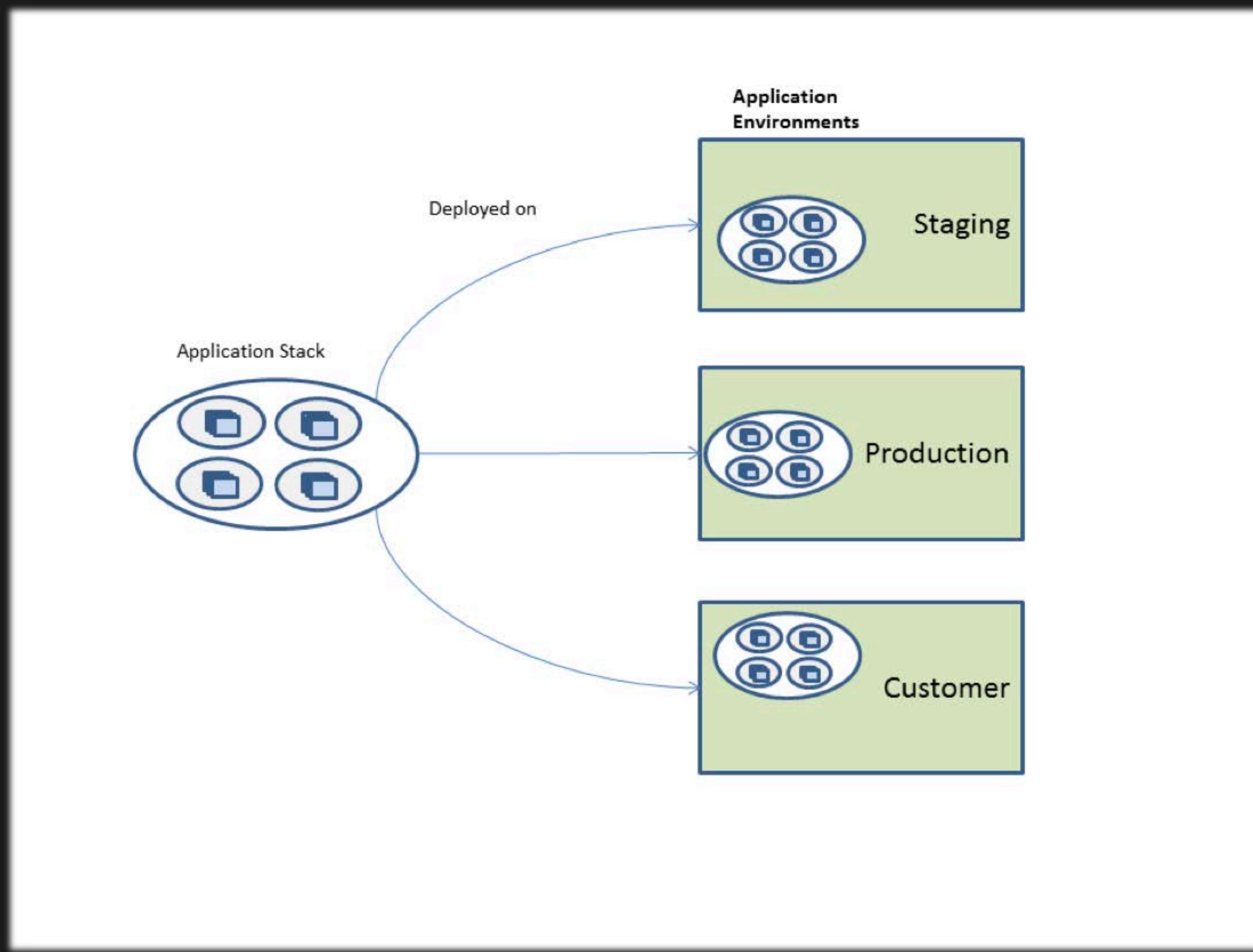
  tags {
    Name = "HelloWorld"
  }
}

resource "aws_rds_cluster" "postgresql" {
  cluster_identifier      = "aurora-cluster-demo"
  engine                  = "aurora-postgresql"
  availability_zones     = ["us-west-2a", "us-west-2b", "us-west-2c"]
  database_name           = "mydb"
  master_username         = "foo"
  master_password         = "bar"
  backup_retention_period = 5
  preferred_backup_window = "07:00-09:00"
}

resource "aws_security_group" "allow_all" {
  name      = "allow_all"
  description = "Allow all inbound traffic"
  vpc_id    = "${aws_vpc.main.id}"

  ingress {
    from_port  = 0
    to_port    = 0
    protocol   = "-1"
  }
}
```

ENVIRONMENT



EC2 Dashboard

- Events
- Tags
- Reports
- Limits
- INSTANCES**
 - Instances
 - Spot Requests
 - Reserved Instances
 - Scheduled Instances
 - Dedicated Hosts
- IMAGES**
 - AMIs
 - Bundle Tasks
- ELASTIC BLOCK STORE**
 - Volumes
 - Snapshots
- NETWORK & SECURITY**
 - Security Groups
 - Elastic IPs
 - Placement Groups
 - Key Pairs
 - Network Interfaces
- LOAD BALANCING**
 - Load Balancers
 - Target Groups
- AUTO SCALING**
 - Launch Configurations
 - Auto Scaling Groups
- SYSTEMS MANAGER SERVICES**
 - Run Command

Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

60 Running Instances

5 Elastic IPs

0 Dedicated Hosts

133 Snapshots

71 Volumes

4 Load Balancers

98 Key Pairs

125 Security Groups

0 Placement Groups

Just need a simple virtual private server? Get everything you need to jumpstart your project - compute, storage, and networking – for a low, predictable price. [Try Amazon Lightsail for free.](#) X

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance



Note: Your instances will launch in the US West (Oregon) region

Service Health

Service Status:

US West (Oregon):

This service is operating normally

Availability Zone Status:

us-west-2a:

Availability zone is operating normally

us-west-2b:

Availability zone is operating normally

us-west-2c:

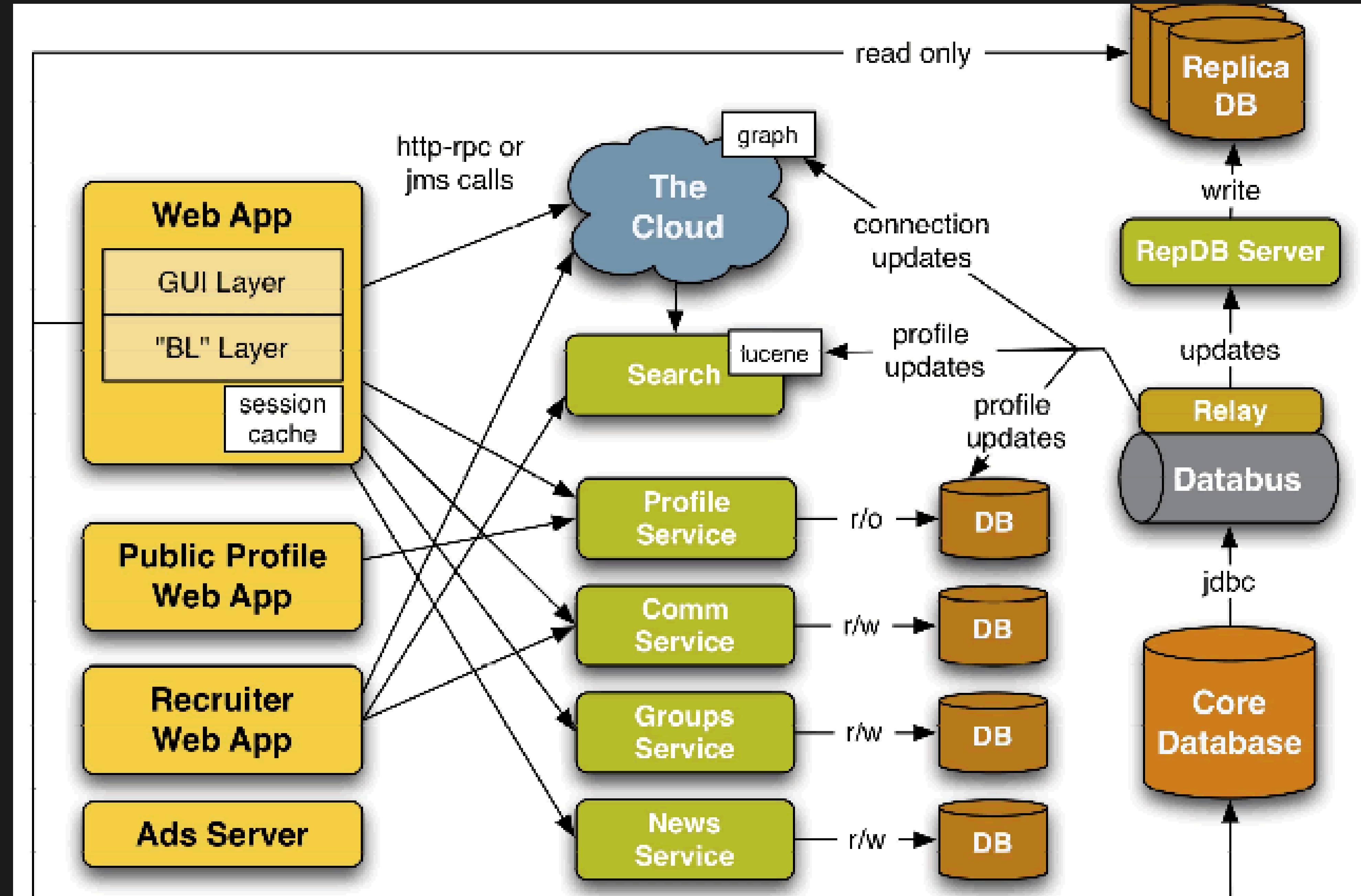
Availability zone is operating normally

[Service Health Dashboard](#)

Scheduled Events

US West (Oregon):

No events



Self-service



- Self-service
- Documentation

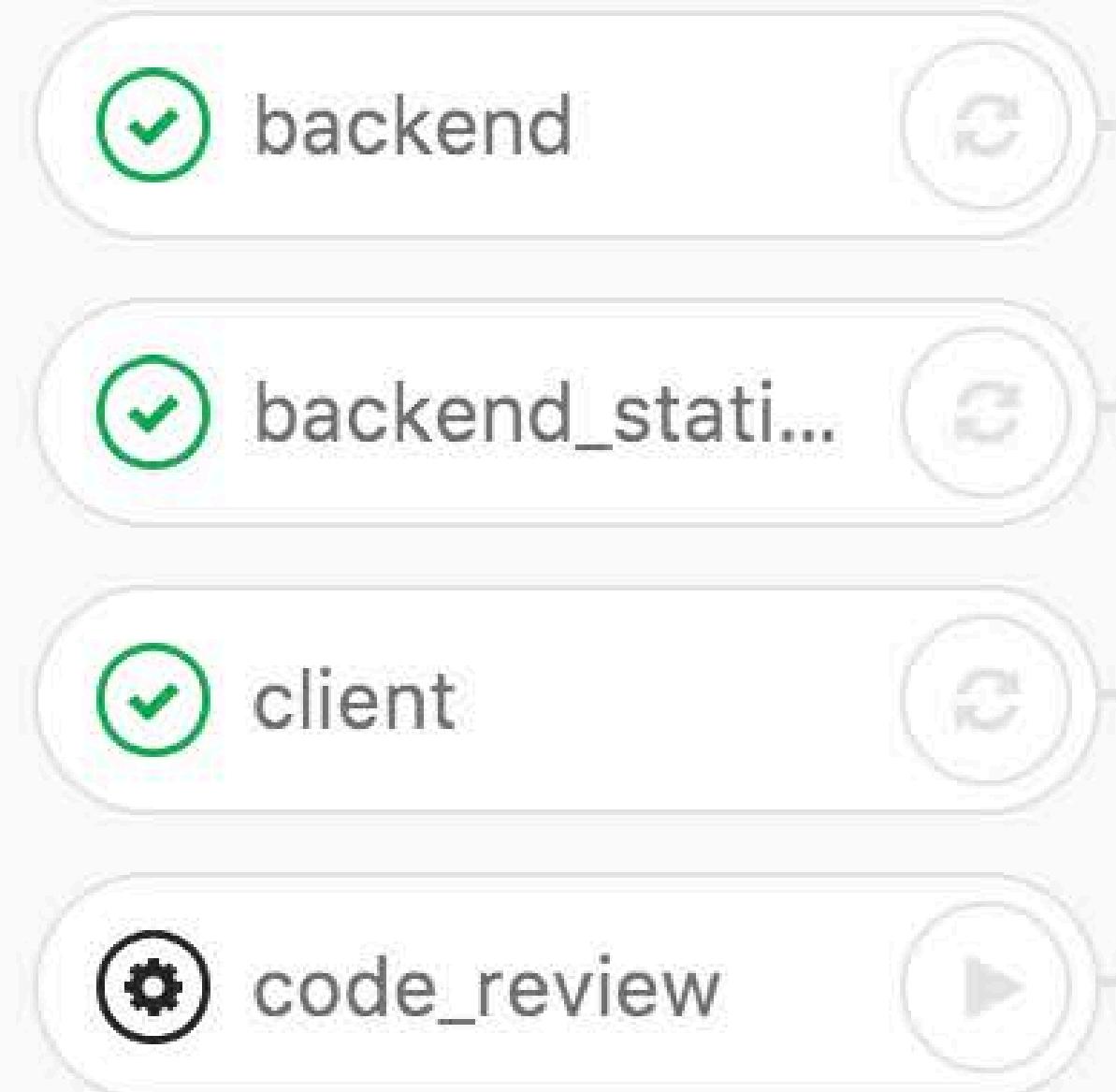




- Self-service
- Documentation
- **Speed and safety**

- Self-service
- Documentation
- Speed and Safety
- Version Control

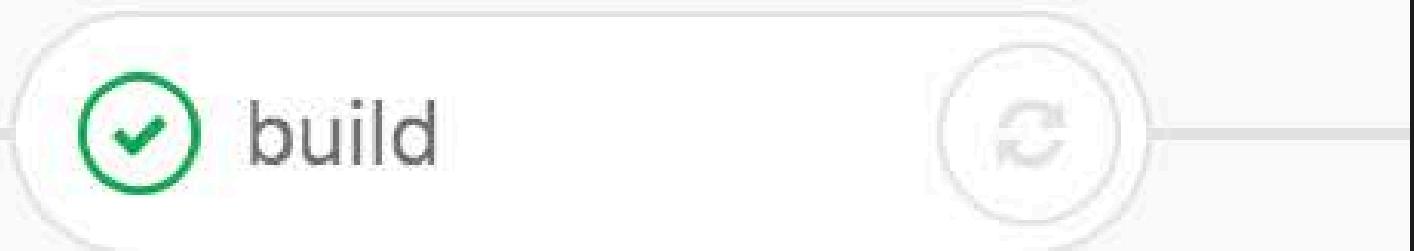
Test



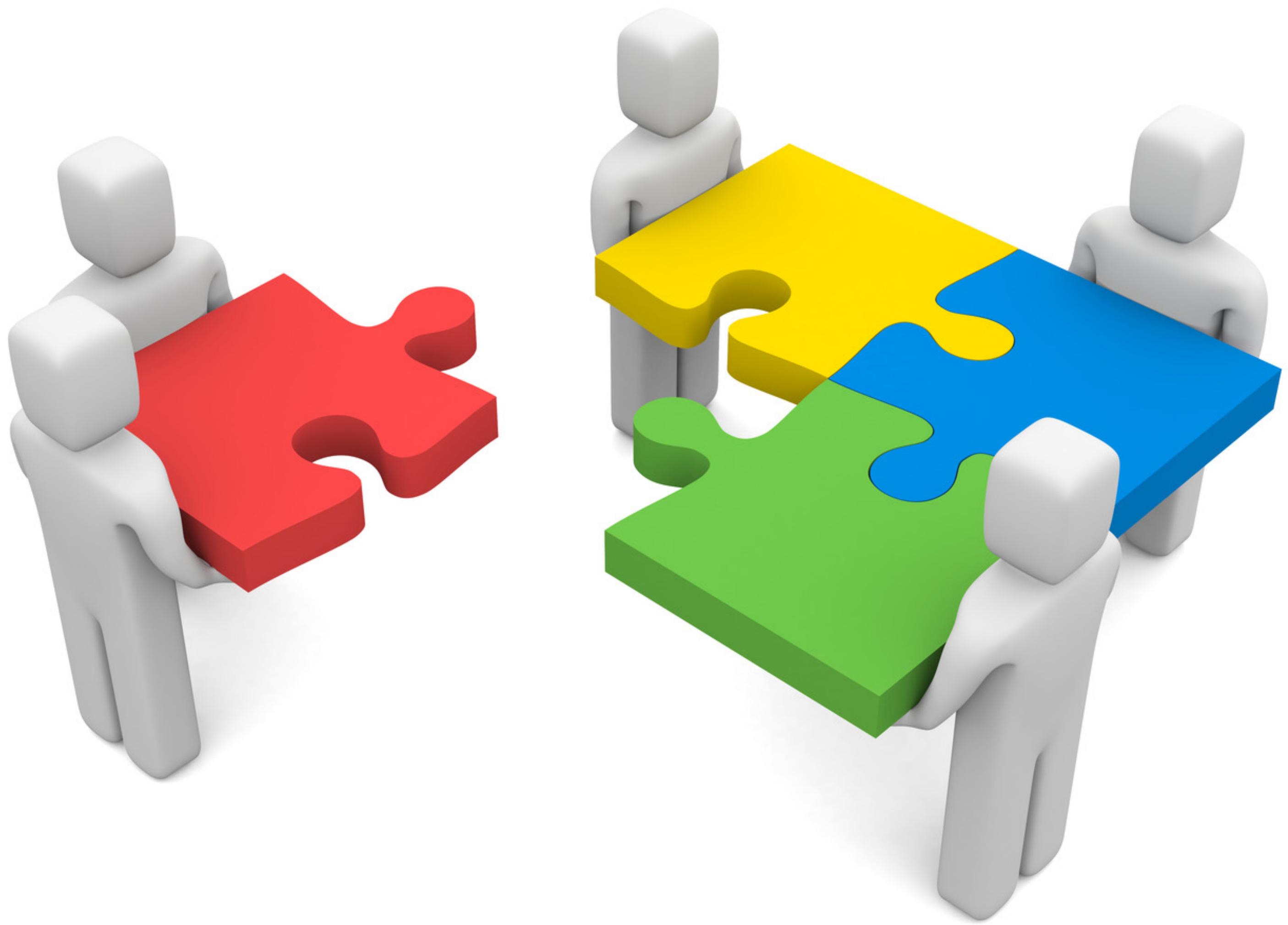
Security



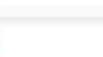
Build



- Self-service
- Documentation
- Speed and Safety
- Version Control
- Validation



- ❑ Self-service
- ❑ Documentation
- ❑ Speed and Safety
- ❑ Version Control
- ❑ Validation
- ❑ Reusability

Actions		Launch Instance	Connect	Actions	?				
	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
<input type="checkbox"/>	i-0d15c1	i-0d15c1	t2.medium	us-west-2a	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-03713f	i-03713f	t2.small	us-west-2a	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0d176f	i-0d176f	t2.small	us-west-2b	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0616d1	i-0616d1	t2.small	us-west-2a	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0c85f9	i-0c85f9	t2.small	us-west-2c	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0d8681	i-0d8681	t2.small	us-west-2b	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-004cd0	i-004cd0	t2.micro	us-west-2a	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0cff32	i-0cff32	t2.medium	us-west-2b	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-03d594	i-03d594	t2.medium	us-west-2b	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0017f8	i-0017f8	t2.micro	us-west-2a	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0d9e91	i-0d9e91	m3.large	us-west-2b	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0c78cc	i-0c78cc	m3.large	us-west-2b	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-017c43	i-017c43	t2.micro	us-west-2b	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-015b50	i-015b50	t2.micro	us-west-2c	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-03a4e1	i-03a4e1	t2.micro	us-west-2c	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-092b21	i-092b21	t2.medium	us-west-2b	● stopped		None	 ec	-
<input type="checkbox"/>	i-016164	i-016164	t2.medium	us-west-2b	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-08bf73	i-08bf73	t2.medium	us-west-2a	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-0ac7ea	i-0ac7ea	t2.medium	us-west-2a	● running	● 2/2 checks ...	None	 ec	-
<input type="checkbox"/>	i-07a05f	i-07a05f	t2.micro	us-west-2a	● running	● 2/2 checks ...	None	 ec	-

- Self-service
- Documentation
- Speed and safety
- Version control
- Validation
- Reusability
- Happiness

- ❑ Self-service
- ❑ Documentation
- ❑ Speed and safety
- ❑ Version control
- ❑ Validation
- ❑ Reusability
- ❑ Happiness

INFRASTRUCTURE AS CODE

Self-service

Documentation

Speed and safety

Version control

Validation

Reusability

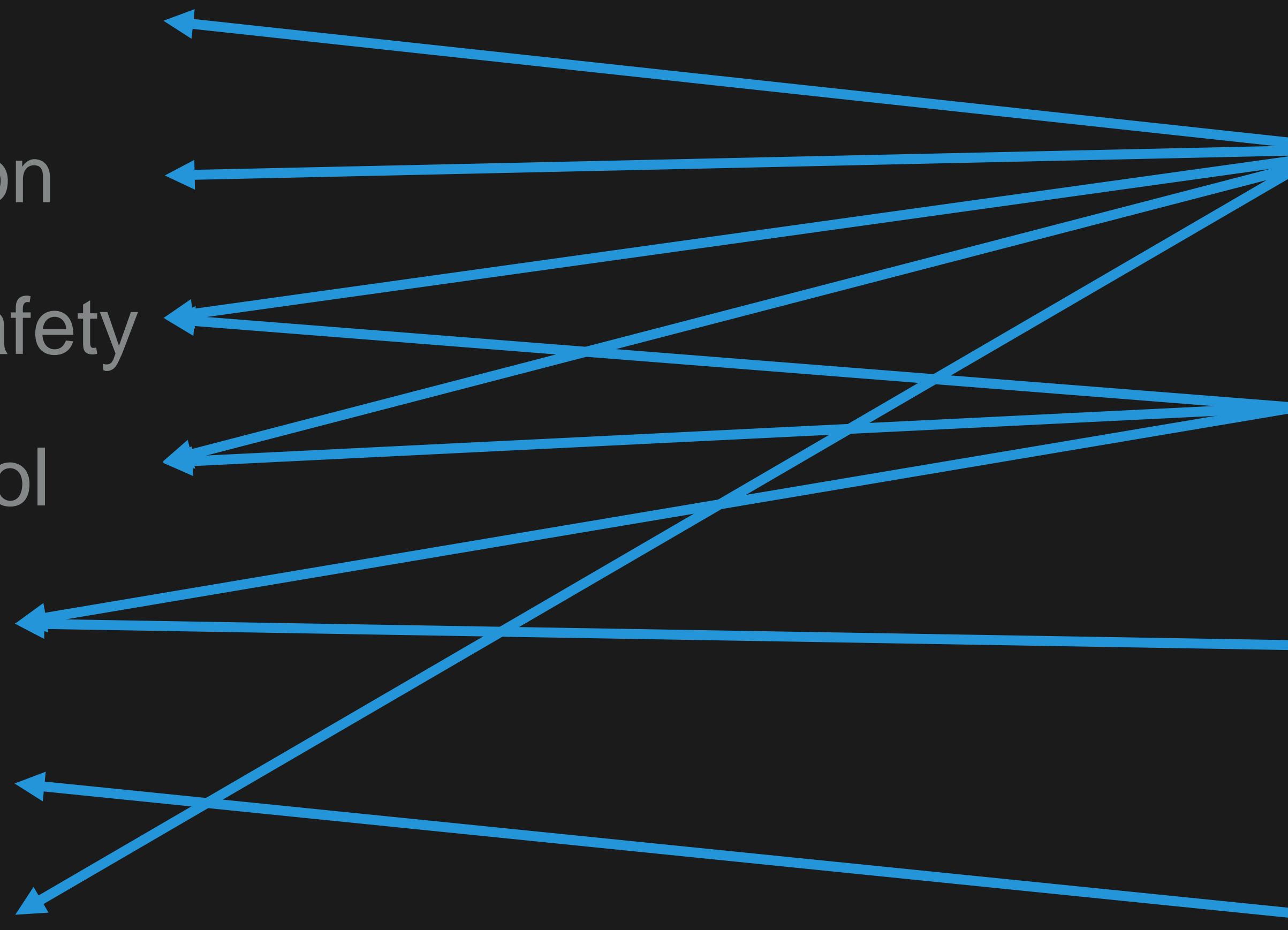
Happiness

Maintainability

Security

Testability

Reusability





CODE IS
POWERFUL

	Source	Cloud Type	Infrastructure	Language	Agent	Master	Community	Maturity	
Chef	Open	All	Config Mgmt	Mutable	Procedural	Yes	Yes	Large	High
Puppet	Open	All	Config Mgmt	Mutable	Declarative	Yes	Yes	Large	High
Ansible	Open	All	Config Mgmt	Mutable	Procedural	No	No	Large	Medium
SaltStack	Open	All	Config Mgmt	Mutable	Declarative	Yes	Yes	Medium	Medium
CloudFormation	Closed	AWS	Provisioning	Immutable	Declarative	No	No	Small	Medium
Heat	Open	All	Provisioning	Immutable	Declarative	No	No	Small	Low
Terraform	Open	All	Provisioning	Immutable	Declarative	No	No	Medium	Low

* Terraform: Up and Running

INTRO TO TERRAFORM

› Providers

- › Alicloud
- › Archive
- › Arukas
- › AWS
- › Bitbucket
- › CenturyLinkCloud
- › Chef
- › Circonus
- › Cloudflare
- › CloudStack
- › Cobbler
- › Consul
- › Datadog
- › DigitalOcean
- › DNS
- › DNSMadeEasy
- › DNSimple
- › Docker
- › Dyn
- › External
- › Fastly
- › GitHub
- › Gitlab
- › Google Cloud
- › Grafana
- › Heroku
- › HTTP
- › Icinga2
- › Ignition

- › InfluxDB
- › Kubernetes
- › Librato
- › Local
- › Logentries
- › Mailgun
- › New Relic
- › Nomad
- › NS1
- › Microsoft Azure
- › Microsoft Azure (Legacy ASM)
- › MySQL
- › 1&1
- › Oracle Public Cloud
- › OpenStack
- › OpsGenie
- › OVH
- › Packet
- › PagerDuty
- › PostgreSQL
- › PowerDNS
- › ProfitBricks
- › RabbitMQ
- › Rancher
- › Random
- › Rundeck
- › Scaleway
- › SoftLayer
- › StatusCake
- › Spotinst
- › Template

› Terraform

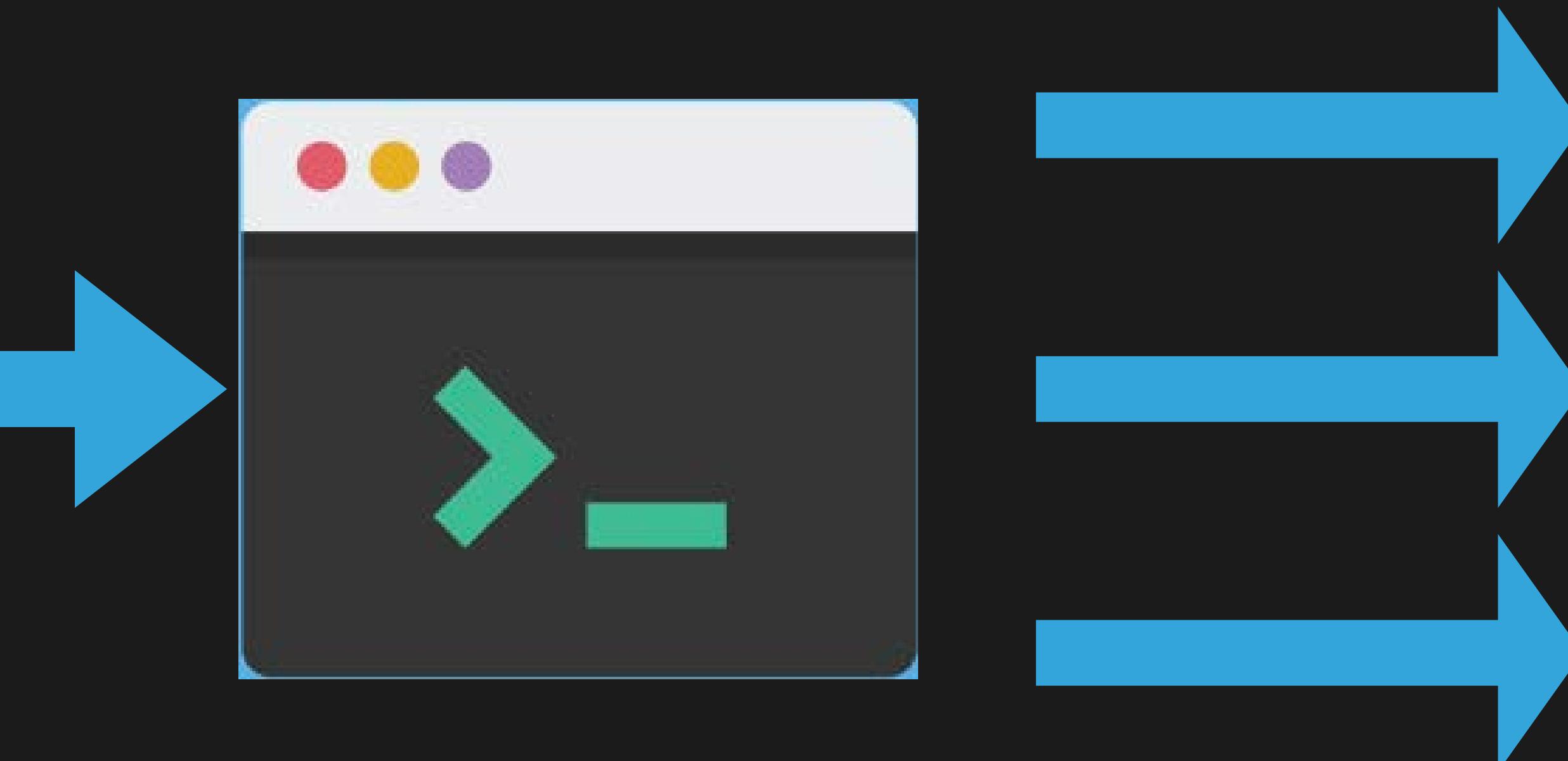
- › Terraform Enterprise
- › TLS
- › Triton
- › UltraDNS
- › Vault
- › VMware vCloud Director
- › VMware vSphere

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "example" {
  ami           = "ami-40d28157"
  instance_type = "t2.micro"

  tags {
    Name = "terraform-example"
  }
}
```

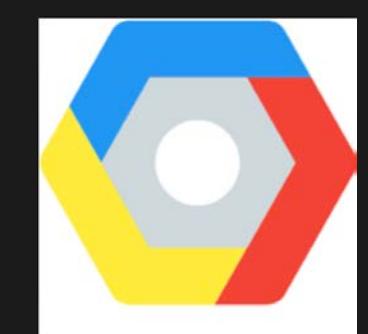
```
provider "aws" {  
  region = "us-east-1"  
}  
  
resource "aws_instance" "example" {  
  ami           = "ami-40d28157"  
  instance_type = "t2.micro"  
  
  tags {  
    Name = "terraform-example"  
  }  
}
```



*.TF FILES

TERRAFORM

API CALLS



O'REILLY®



Yevgeniy Brikman

Source Code

https://github.com/arturopie/iac_with_terraform



CODE IS
POWERFUL

THANK YOU

Code is powerful

Q&A

Arturo Pie - @arturo_pie

Source Code

https://github.com/arturopie/iac_with_terraform

Code is powerful

Q&A

Arturo Pie - @arturo_pie

Source Code

https://github.com/arturopie/iac_with_terraform

NO SILVER BULLET

Maintainability

Security

Testability

Reusability

What is Terraform & Infrastructure as Code (IaC)?

If you’re just getting started in software engineering, or you’ve been around a long time, you’ve probably at least heard the terms “Infrastructure as Code” and “Terraform” mentioned. But what are they, and why are they important?

Terraform is an Infrastructure as Code (IaC) tool that allows engineers to define their software infrastructure in code. While the idea of “code” may not be novel to engineers; the ability to provision infrastructure this way is a powerful abstraction that enables managing large distributed systems at scale.

In this article, we’ll take a look at what Infrastructure as Code and Terraform are, how they can help you in your work as a developer, and how you can get started using them.

Grab the Terraform cheat sheet

Check out the top 10 Terraform commands and get a full rundown of all the basic commands you need to get the most out of Terraform in our [Terraform cheat sheet](#).

What is Infrastructure as Code?

Infrastructure as Code is a way of defining and managing your infrastructure using code, rather than manual processes like clicking through a UI or using the command line. This means that you can manage your infrastructure in the same way that you manage your application code – with version control, automation, and collaboration. In other words, infrastructure as code is a way of making your infrastructure more like software.

Historically, Infrastructure as Code has seen many iterations, starting with configuration management tools like [CFEngine](#), [Chef](#), [Puppet](#), [Ansible](#), and [Salt](#). Newer tooling like [Cloudformation](#) and [Terraform](#) take a declarative approach and focus on the actual provisioning of resources, as opposed to the configuration of existing ones. The newest generation of tools focuses on using the capabilities of existing imperative programming languages. [AWS](#) and [Terraform](#) both provide Cloud Development Kits(CDKs), and [Pulumi](#) is also a popular option for provisioning infrastructure with traditional software tools.

What is Terraform?

Terraform is a tool for provisioning, managing, and deploying infrastructure resources. It is an open-source tool written in Golang and created by [the HashiCorp company](#). With Terraform, you can manage infrastructure for your applications across multiple cloud providers – AWS, Azure, GCP, etc. – using a single tool.

To get started with Terraform, developers simply need to download the Terraform binary, choose which [provider/platform](#) they’ll be working with, create some [boilerplate configuration for that provider](#), and they can get started creating infrastructure code.

One of the key features of Terraform is its declarative syntax. This means that you define what your desired end state is, and Terraform figures out the best way to achieve that. Compared to the imperative workflow of traditional programming languages, this can be a bit of a shift in mindset – but it enables managing infrastructure deployments at scale without the steep learning curve typical to software development.

The typical workflow for provisioning resources with Terraform is as follows:

1. Some Terraform configuration is written, including the provider definition.
2. The working directory is initialized(this is called the root module).
3. Provider plugins are downloaded.
4. The command `terraform plan` is run in the root module, generating a proposed plan to provision resources.
5. If the plan is acceptable, the `terraform apply` command is run and resources are provisioned.

Terraform keeps track of the state of the resources it manages in a [state file](#). This file is essentially a large JSON data structure that tracks proposed changes to infrastructure, as well as out-of-band changes that may have occurred to live resources outside of the Terraform configuration. New Terraform users typically maintain a local state file on their workstation or laptop. At scale with multiple engineers managing infrastructure, the state is typically broken down into multiple files and stored remotely using services like AWS S3.

Terraform is also “idempotent”, which means that repeated plan/apply cycles will not trigger a re-deploy of resources; only changes to the existing state will be reflected in a new plan and apply invocation.

Now that you’ve learned a little bit about what Terraform is and what it does, we can start to explore the “why” of Terraform and the benefits it provides.

What are the benefits of Terraform?

Utilizing Infrastructure as Code to manage and deploy infrastructure resources [unlocks several benefits for developers and engineers](#). Making Terraform your tool of choice for IaC confers additional benefits that allow developers to leverage easy-to-use tools to manage complex software architecture.

1. Terraform configuration is written using a declarative paradigm.

The best way to understand declarative code is to compare it to the more familiar pattern of imperative logic that most modern programming languages use.

Imagine a basic Python program that takes a series of numbers as input from the user, prints each number out in ascending order, then returns the sum of all the inputs. A programmer needs to write the code in a specific, logical order, or the program will fail. If the program attempts to sum all the numbers before the input is received, it is likely to generate some kind of exception or error. If the programmer wants the numbers to be sorted into the correct order, that will need to occur before they are printed as output.

In each part of the program, the programmer is having to specifically define both the logic and order of logic when the program executes. In contrast, declarative program syntax means the programmer needs to only “declare” the desired end state of the program. The compiler, in this case, the Terraform binary, is programmed to determine the best order of operations path through which they achieve the desired end state described in the configuration. This is especially helpful in dealing with cloud provider APIs, as many cloud resources have dependencies on the creation of other foundational resources before creation can proceed.

2. Hashicorp Configuration Language (HCL) is a Domain Specific Language (DSL).

Domain-specific languages are designed with a specific use case in mind, specialized to handle the requirements and constraints of a specific application or program domain.

If the utility of a DSL doesn’t seem immediately obvious, you should consider one of the most famous and widely used DSLs: HTML. HTML is a markup language that focuses on the domain of hyper-text (read: the internet). HTML does away with complex program logic and syntax and focuses on the specific use-case of content presentation.

In the case of Terraform, developers only need to learn a minimal amount of HCL syntax before they can be productive. As a DSL, Terraform makes development easier and more efficient by abstracting away the complexity inherent in general-purpose languages. That’s not to say that Terraform doesn’t have more complex logic; advanced users can use newer, imperative constructs like for-loops and if/then logic.

HCL is considered a superset of the [JSON language](#), which means it shares similar syntax, but also has additional features beyond the scope of JSON.

3. Terraform is widely adopted

Terraform is generally considered the industry standard when it comes to Infrastructure as Code tooling. It isn’t always good advice to go with the herd, but when it comes to technology implementation, choosing a tool with a large community, solid support base, and multi-year longevity is critically important. No one wants to have to explain to stakeholders and customers that the application is down because the code or platform is no longer supported!

The other benefit of wide adoption is the collective, shared knowledge of the community. Best practices are developed and shared, and an ecosystem of supporting tools and documentation can be built. A great example of the power of community support is the [awesome-terraform repository on Github](#), a curated list of tools, libraries, documentation, blogs, and more.

4. Terraform enables immutability

Immutable infrastructure makes managing complex distributed systems easier and safer and allows them to scale much more reliably. What exactly is meant by “immutable infrastructure”, and how does Terraform enable it?

Consider a hypothetical scenario: a developer needs to deploy some changes to fix a production application. They create their changes locally, run some tests and linting to validate the changes and check syntax, and now they’re ready to deploy. However, to get their code to run in the staging environment, they have to change some configuration to point to the staging database. Then they

need to give the QA team access to the staging servers. Everything manages to check out in the staging environment, but when deploying to production disaster strikes, the application stops serving traffic and an outage occurs.

Was the original code bad? Or was it the changes made in staging? Because the lines between environments and stages were blurred, it's nearly impossible to tell. Mutable infrastructure means changes can occur at any point in the lifecycle of an application or its infrastructure.

With immutable infrastructure, build, release, and deploy stages are kept separate. Once code changes are built, they are stamped with an immutable release tag. Further changes or fixes result in a new tag being generated. Developers and engineers know that a change that was made in a local development environment is the same change across different environments and deployment stages.

[The 12-factor app framework highlights this pattern in factor V.](#)

5. Terraform is modular

Modularity is an important feature in a variety of languages and systems. Abstracting logic and resources behind simple interfaces is one of the best ways to manage complexity at scale. As Terraform deployments grow more complex, developers can consider employing [modules](#) to encapsulate various resources in a reusable package.

A common use case is providing other development teams with Kubernetes clusters for testing and development. Normally, provisioning a Kubernetes cluster in a cloud provider like AWS requires a lot of boilerplate configuration and resources, even when using managed services. With modules, that configuration can be hidden behind a basic configuration interface. Developers can import the module, specify whatever inputs the module author has provided, and they can provision a complete stack without having to duplicate effort needlessly.

How can I use Terraform?

Although it's simple to get started, Terraform is a very powerful tool for provisioning infrastructure. The key for new users is to start small with basic configuration, and work towards fully automating their infrastructure as code.

New developers should start with something simple; a couple of basic resources without complex dependency chains. You don't need to start by trying to manage your entire production application environment on day 1: try using Terraform to manage the S3 bucket where deployment artifacts are stored, or Google DNS records for a frontend website.

Once you're comfortable, you can start to iterate and grow your Terraform usage. One of the first steps is to move from local state to a remote state file with locking. It's virtually impossible to safely manage larger Terraform deployments with multiple users without the state having a locking mechanism. When a Terraform plan or apply is running, locks prevent other users from making changes that could result in an inconsistent state or corruption.

With multiple users now contributing Terraform configuration, you can start to expand usage to cover more and more of your infrastructure resources, including networking, security, CDN, and more. As your infrastructure increases in complexity, consider encapsulating certain segments of your architecture in modules.

Finally, your team can start to use Terraform to provision resources as part of your CI/CD strategy. CI/CD is a complex topic that we won't cover here, but [GitLab](#) and [GitHub](#) both provide great batteries-included solutions for deployment automation that can be used with Terraform. Hashicorp [provides solid](#) documentation specifically targeted at users who are considering automating their Terraform deployments.

Conclusion

Why provision infrastructure with clicks and manual processes while writing application code? Infrastructure as Code tools like Terraform means that infrastructure configuration can be brought into the same development processes, allowing for testing, standardization, and scalability. The modern Infrastructure as Code ecosystem has a broad variety of resources for learning and getting started.

Want to learn more about Terraform?

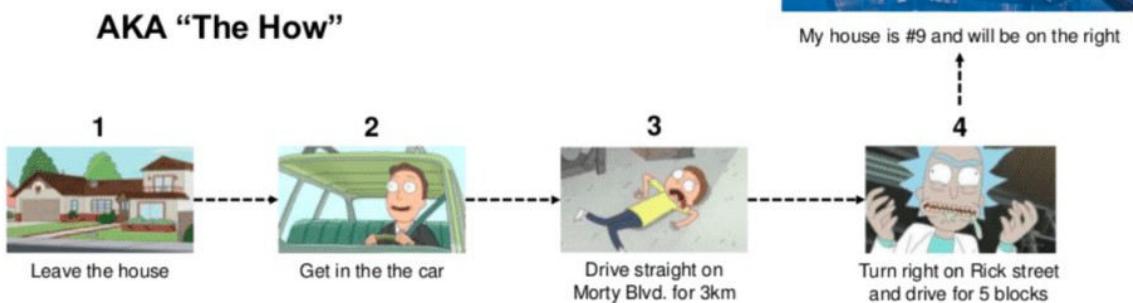
Why not try studying for the Hashicorp Certified: Terraform associate certification? Pluralsight offers an [excellent cert prep course](#) which teaches you everything you need to know about Terraform, including how to use it effectively in your own projects. It's a great way to learn about Terraform, even if you don't end up taking the certification exam.

Infrastructure as Code (IaC): Declarative vs Imperative

IaC: Imperative

Imperative (procedural):

Defines [specific commands](#) that need to be executed in the appropriate order to end with the desired conclusion.



Imperative – focus on the **actual provisioning process** and may reference a file containing a list of settings and configuration values.

Declarative – focused on the **desired end state** of deployment and relies on an interpretation engine to create and configure the actual resources.

When to use IaC

- You use a large amount of IaaS resources.
- Your infrastructure is rented from many different providers or platforms.

- You need to make regular adjustments to your infrastructure.
- You need proper documentation of changes made to your infrastructure.
- You want to optimize collaboration between administrators and developers.

Advantages of IaC

- Minimizing shadow IT inside of businesses and enabling quick, effective infrastructure upgrades made concurrently with application development
- Creating trackable and auditable configurations by enabling version-controlled infrastructure and configuration modifications.
- Maintaining infrastructure and settings in the appropriate condition while effectively managing configuration drift.
- Connecting directly to platforms for CI/CD.

Declarative (functional):
Defines the **desired state** and the system executes what needs to happen to achieve that desired state.

AKA “The What”



IaC Tools & Platforms – Overview

K2IAcademy

DEVOPS IAC TOOLS OVERVIEW

Terraform

The top infrastructure as code (IaC) solution for managing infrastructure across a variety of clouds, including AWS, Azure, GCP, Oracle Cloud, Alibaba Cloud, and even platforms like Kubernetes and Heroku, is [Terraform](#) by HashiCorp.



While assuring the intended state throughout the settings, Terraform may be utilized to facilitate any infrastructure provisioning and management use cases across many platforms and providers.

Ansible

[Ansible](#) is a free, open-source configuration management solution with IaC capabilities rather than a specialized infrastructure management tool. It is an agentless solution that works with both cloud and on-premises systems and may be used by SSH or WinRM.



It has limitations when it comes to maintaining that infrastructure but shines at provisioning infrastructure and configuration management.

Chef/Puppet

There are two effective configuration management tools: Chef and Puppet. Both seek to offer infrastructure management skills with configuration management and automation across the development process.

1. Chef was created with stronger collaboration features to be readily integrated into DevOps methods. It is used for configuration management and is Best used for Deploying and

configuring applications using a pull-based approach.



2. Puppet developed as a result of focusing on pure process automation. Currently, Puppet has automatic observers built in to detect configuration drift. It is a popular tool for configuration management and needs agents to be deployed on the target machines before



puppet can start managing them.

Tools Overview Table:

Tool	Tool Type	Infrastructure	Architecture	Approach	Manifest Written Language
puppet	Configuration Management	Mutable	Pull	Declarative	Domain Specific Language (DSL) & Embedded Ruby (ERB)
CHEF	Configuration Management	Mutable	Pull	Declarative & Imperative	Ruby
ANSIBLE	Configuration Management	Mutable	Push	Declarative & Imperative	YAML
SALTSTACK	Configuration Management	Mutable	Push & Pull	Declarative & Imperative	YAML
Terraform	Provisioning	Immutable	Push	Declarative	HashiCorp Configuration Language (HCL)

IaC DevOps Best Practices

Avoid automating everything right away

- Try not to automate everything right away if you are a company, an application, or a platform that is still in the early stages of development. This is due to the potential for rapid

change. You may start automating your platform's provisioning and maintenance after it has more or less reached a stable state.

Check and keep an eye on your setups

- Infrastructure as Code should and can still be tested since it is still code. Before deploying your servers, you should install testing and monitoring tools for IaC to look for flaws and inconsistencies.

The more rigid the better

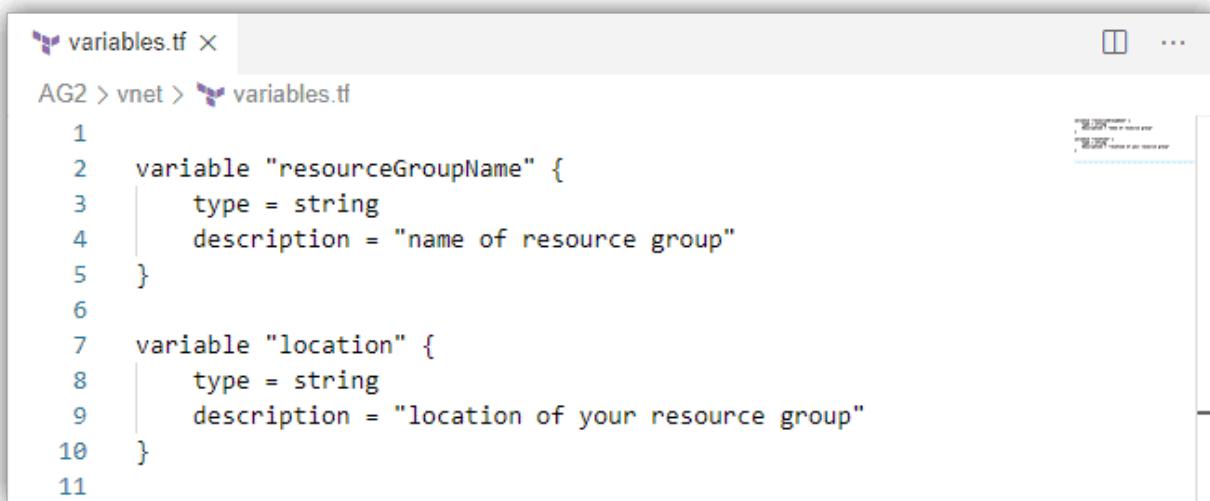
- Give as much detail as you can about the atmosphere you wish to create. Include the developers in the creation of the IaC standards for the runtime environments and infrastructure components. Making sure there are no flaws in the code can make it operate more efficiently.

Terraform Variables: Terraform Variable Types and Uses for Beginners

Terraform Input Variables

Terraform input variables are used as parameters to input values at run time to customize our deployments. Input terraform variables can be defined in the main.tf configuration file but it is a best practice to define them in a separate variable.tf file to provide better readability and organization.

A variable is defined by using a **variable** block with a label. The label is the name of the variable and must be unique among all the variables in the same configuration.



```
variables.tf
AG2 > vnet > variables.tf
1
2   variable "resourceGroupName" {
3     type = string
4     description = "name of resource group"
5   }
6
7   variable "location" {
8     type = string
9     description = "location of your resource group"
10  }
11
```

The variable declaration can optionally include three arguments:

- **description:** briefly explain the purpose of the variable and what kind of value is expected.
- **type:** specifies the type of value such as string, number, bool, map, list, etc.
- **default:** If present, the variable is considered to be optional and if no value is set, the default value is used.

Check out: How to [Install Terraform](#) in Linux, Mac, Windows

Terraform Input Variables

The type argument in a variable block allows you to enforce [type constraints](#) on the variables a user passes in. Terraform supports a number of types, including **string**, **number**, **bool**, **list**, **map**, **set**, **object**, **tuple**, and **any**.

If a type isn't specified, then Terraform assumes the type is any. Now, let's have a look at some of these terraform variables types and their classification.

Primitive Types

A *primitive* type is a simple type that isn't made from any other type. The available primitive types are:

- **string:** a sequence of characters representing some text, such as “hello”.
- **number:** a numeric value. The number type can represent both whole numbers like 15 and fractional values such as 6.28318.
- **bool:** either true or false.

Also Read: [Terraform Workflow](#).

Complex Types

A *complex* type is a type that groups multiple values into a single value. These values could be of a similar type or different types.

1. **List:** A Terraform list variable is a sequence of similar values indexed by numbers (starting with 0). It accepts any type of value as long as they are all of the same types. Lists can be defined either implicitly or explicitly.

```
# implicitly by using brackets [...]
variable "cidrs" { default = [] }

# explicitly
variable "cidrs" { type = "list" }
```

2. **Map:** A map is a collection of values where each value is identified by a string label. In the example below is a map variable named **managed_disk_type** to define the type of storage we want to use based on the region in Azure. In the default block, there are two string values, “Premium_LRS” and “Standard_LRS” with a named label to identify each one

westus2 and eastus.

```
variable "managed_disk_type" {
  type = map
  description = "Disk type Premium in Primary location Standard in DR location"

  default = {
    westus2 = "Premium_LRS"
    eastus = "Standard_LRS"
  }
}
```

3. **Object:** An object is a structural type that can contain different types of values, unlike map, list. It is a collection of named attributes that each have their own type.

In the below example, we have declared an object type variable **os** for the os image that can be used to deploy a VM.

```
variable "os" {
  description = "OS image to deploy"
  type = object({
    publisher = string
    offer = string
    sku = string
    version = string
  })
}
```

Lists, maps, and objects are the three most common complex variable types. They all can be used for their specific use cases.

Also read: Step-by-step guide on [Terraform Certification](#)

Use Input Variables

After declaration, we can use the variables in our main.tf file by calling that variable in the **var.<name>** format. The value to the variables is assigned during **terraform apply**.

```
main.tf
provider "azurerm" {
  version = "1.38.0"
}

#Create resource group
resource "azurerm_resource_group" "TFAzure-rg" {
  name      = var.resourceGroupName
  location  = var.location
  tags      = {
    Environment = "Dev"
  }
}
```

Also Check Our blog post on [Terraform Cheat Sheet](#). Click here

Assign Values To Input Variables

There are multiple ways to assign values to variables. The following is the descending order of precedence in which variables are considered.

1. Command-line flags

The most simple way to assign value to a variable is using the -var option in the command line when running the terraform plan and terraform apply commands.

```
$ terraform apply -var="resourceGroupName=terraformdemo-rg" -  
var="location=eastus"
```

2. Variable Definition (.tfvars) Files

If there are many variable values to input, we can define them in a variable definition file. Terraform also automatically loads a number of variable definitions files if they are present:

- Files named exactly **terraform.tfvars** or **terraform.tfvars.json**
- Any files with names ending in **.auto.tfvars** or **.auto.tfvars.json**

```
terrafrom.tfvars
resourceGroupName = "terraformdemo"
location = "eastus"
```

If the file is named something else, then use the **-var-file** flag directly to specify a file.

```
$ terraform apply -var-file="testing.tfvars"
```

3. Terraform Environment Variables

Terraform searches the environment of its own process for environment variables named **TF_VAR_<var-name>** followed by the name of a declared variable. Terraform scans all variables starting with TF_VAR and uses those as variable values for Terraform.

```
$ export TF_VAR_location=eastus
```

This can be useful when running Terraform in automation, or when running a sequence of Terraform commands in succession with the same terraform variables.

Read More: About [Hashicorp Terraform](#). Click here

Define Output Variables

Outputs allow us to define values in the configuration that we want to share with other resources or modules. For example, we could pass on the output information for the public IP address of a server to another process.

An output variable is defined by using an **output** block with a label. The label must be unique as it can be used to reference the output's value. Let's define an output to show us the public IP address of the server. Add this to any of the *.tf files.

```
output "public_ip_address" {
  description = "Public IP Address of Virtual Machine"
  value = azurerm_public_ip.publicip.ip_address
}
```

Multiple output

blocks can be defined to specify multiple output variables. Outputs are only shown when Terraform applies your plan, running a terraform plan will not render any outputs.

How to Provision AWS Infrastructure with Terraform?

Cloud itself is a big domain with various services running simultaneously. It will require a lot of effort for a person/organization to manage the cloud without automation. Thus, cloud automation is on its pace and various tools are proposed for faster and efficient development. One such automation tool is Terraform.

Prepare Your System

In this tutorial, we will use the Amazon EC2 instance with the Ubuntu system to run Terraform. But, it's your choice to run Terraform on any cloud platform or machine. For a better understanding, I will recommend you to follow the steps with us.

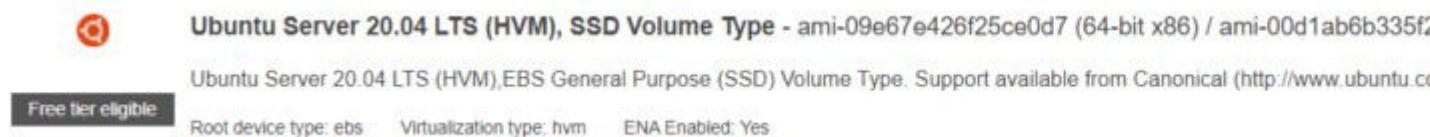
Step 1) Open up your [AWS console](#) or directly visit ‘console.aws.amazon.com‘. If you don’t have access to AWS, [create one free tier account](#).

Step 2) Search for EC2 in your AWS console and open it.

Step 3) Click on Launch Instances to create a new EC2 instance.



Step 4) Select an AMI (Amazon Machine Image). In our case, we will use Ubuntu.



Step 5) You can fill up all the details for your instance. To quickly create an EC2 instance, leave the settings to default, launch your instance and save your new RSA key pair safe.

	Name	Instance ID	Instance state	Instance type
<input type="checkbox"/>	TerraformDemo	i-019be03da0a8d753c	Running	t2.micro

Step 6) After a few minutes, your instance will be live and running, as shown in the image above. Select it and click on connect to launch your EC2 instance.

Download, Install and Start Terraform

Based on your system requirement, you can download and install Terraform from the [official download page](#).

Step 1) From the official Terraform download page, copy the link to the Linux file as shown in the image below.



macOS
64-bit | Arm64



FreeBSD
32-bit | 64-bit | Arm



Linux
32-bit | 64-bit | Arm | Arm64



OpenBSD
32-bit | 64-bit



Solaris
64-bit



Windows
32-bit | 64-bit

Step 2) To download Terraform to your AWS instance, use ‘wget’ command followed by the copied download URL as shown in the image below.

```
wget  
https://releases.hashicorp.com/terraform/1.0.9/terraform_1.0.9_linux_amd64.zip
```

```
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.
```

```
ubuntu@ip-172-31-94-180:~$ wget https://releases.hashicorp.com/terraform/1.0.9/terraform_1.0.9_linux_amd64.zip
```

Step 3) To unzip the downloaded package, you need to install unzip tool in your system. Use the below command to download the tool.

```
sudo apt-get install unzip
```

Step 4) Check the downloaded package name using the ‘ls‘ command. Copy the package name and use the unzip command with the package name, as shown below.

```
ls  
unzip terraform_1.0.9_linux_amd64.zip
```

```
ubuntu@ip-172-31-94-180:~$ ls
terraform_1.0.9_linux_amd64.zip
ubuntu@ip-172-31-94-180:~$ unzip terraform_1.0.9_linux_amd64.zip
Archive:  terraform_1.0.9_linux_amd64.zip
  inflating: terraform
ubuntu@ip-172-31-94-180:~$
```

Step 5) Now, the Terraform is extracted successfully. Run the below commands one by one to run the terraform commands hassle-free by ignoring the directories. Finally, use the command ‘`terraform`’ to activate terraform, as shown in the image below.

```
echo $"export PATH=\$PATH:$pwd)" >> ~/.bash_profile
source ~/.bash_profile

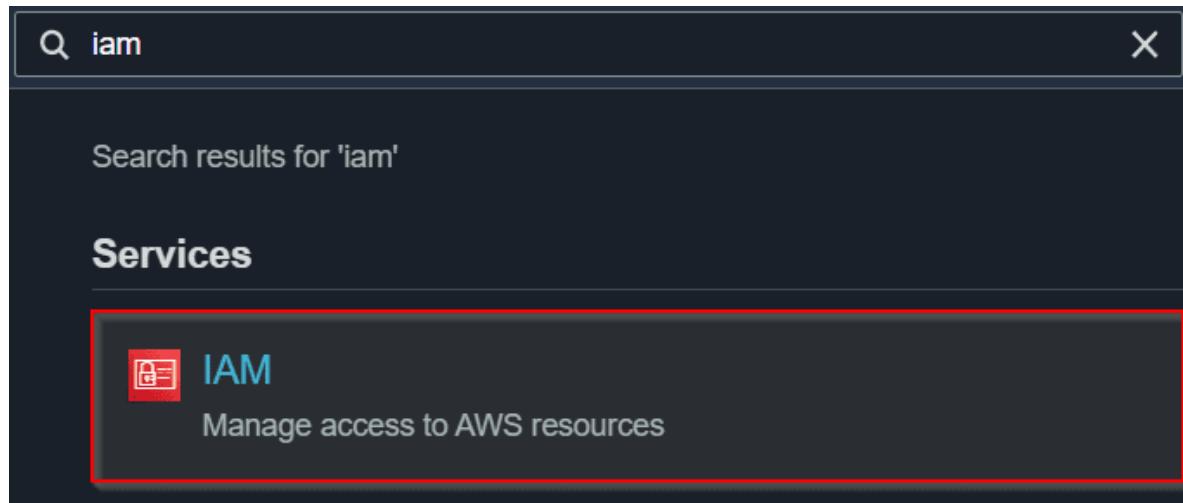
terraform
```

```
ubuntu@ip-172-31-94-180:~$ echo $"export PATH=\$PATH:$pwd)" >> ~/.bash_profile
ubuntu@ip-172-31-94-180:~$
ubuntu@ip-172-31-94-180:~$ source ~/.bash_profile
ubuntu@ip-172-31-94-180:~$
ubuntu@ip-172-31-94-180:~$ terraform
```

Generate AWS IAM Access Key

We installed terraform in our system, but now the question arises: How Terraform will provision anything without AWS permission. For this, we create an access key that will be used for resource provisioning by Terraform.

Step 1) Search and visit IAM in your AWS console.



Step 2) Select *Users* and *Add Users* here if you don't have one, as shown in the image below.

The screenshot shows the AWS IAM 'Users' page. On the left sidebar, under 'Access management', the 'Users' option is selected and highlighted with a red box. At the top right, there are 'Delete' and 'Add users' buttons; the 'Add users' button is also highlighted with a red box. The main area displays a table with columns for 'User name', 'Groups', 'Last activity', 'MFA', 'Password age', and 'Active key age'. A search bar at the top says 'Q, Find users by username or access key'. Below the table, it says 'No resources to display'.

Step 3) Name the user and select Access Key in the checkbox below. Now go to the Next window that is permission settings.

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

The form has a 'User name*' field containing 'terraformdemo', which is highlighted with a red box. Below it is a blue '+ Add another user' button.

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Select AWS credential type*** **Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Step 4) Here, create one new user group.



Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group

[Create group](#)

Step 5) Name the group and search for required permission for allowing Terraform to perform its task. In our case, we will be creating one EC2 instance, so we selected *AmazonEC2FullAccess* permission, as shown in the image below. Now to generate this user group, click on the button Create.

Policy name	Type	Used as	Description
AmazonEC2FullAccess	AWS managed	Permissions policy (1), Boundary (1)	Provides full access to Amazon EC2 via the AWS Management Console.

Step 6) Select the created group for the IAM user and proceed with the final steps to make the new user active.

Group	Attached policies
terraformdemo	AmazonEC2FullAccess

Step 7) After creating the user, you will see the keys on your screen or visit the user to find them. Copy your Access Key ID and Secret Access Key as they will be used while provisioning AWS resources using Terraform.

User	Access key ID	Secret access key
terraformdemo	AKIAVDDOMWVWVZCAGFPXXNK	ofiE6ansggIiXh0II3qHkwJfXz8GgmgQBfl5CNL Hide

Create EC2 Instance with Terraform

For an easy understanding, we will create one EC2 instance using the terraform file with all the instructions to create the EC2 instance.

Step 1) Create a new directory using the ‘mkdir’ command and name it whatever you want. Then, visit the directory using the below commands.

```
mkdir terraform-lab
```

```
cd terraform-lab/
```

Step 2) Create a new file here that will have the instruction to provision the AWS resource. Use the below commands to create the file.

```
vim ec2.tf
```

Step 3) After running the above command, the file is created but to enable the edit mode, you need to press the INSERT or any other button on your keyboard. Copy the below code, replace it with your IAM keys and paste it on your file. Also, don't forget to maintain proper spacing and lines in your code, as shown in the image below.

The provider section in code uses the credentials but if you have an AWS CLI setup you may not require this. The resource section in code will create the resource *aws-instance* with the name *example* with mentioned *AMI (Amazon Machine Image)*.

```
provider "aws" {
  access_key = "ACCESS_KEY_HERE"
  secret_key = "SECRET_KEY_HERE"
  region     = "us-east-1"
}

resource "aws_instance" "example" {
  ami          = "ami-2757f631"
  instance_type = "t2.micro"
}
```

```
provider "aws" {
  access_key = "AKIAVDDGJWJZLAFPXXNK"
  secret_key = "ofiE6ansgqljXh1U2gtkxwJfxz0DqngQBfI5CNL"
  region     = "us-east-1"
}

resource "aws_instance" "example" {
  ami = "ami-2757f631"
  instance_type = "t2.micro"
}
```

Step 4) For saving the above file in the Ubuntu system, press the ‘*Esc*‘ button on your keyboard, type ‘:wq’ and press ‘*Enter*‘.

Step 5) Now everything is ready, use the below command to initialize terraform to compile the file. You will receive an error here if there is any syntax error in the file.

```
terraform init
```

```
ubuntu@ip-172-31-94-180:~/terraform-lab$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.63.0...
- Installed hashicorp/aws v3.63.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
ubuntu@ip-172-31-94-180:~/terraform-lab$ █
```

Step 6) Finally, proceed with resource provisioning with the below command. You will receive an error here if the access keys are wrong, AMI (Amazon Machine Image) is wrong, or permission to provision EC2 Instance is not provided in the IAM (Identity Access Management).

If everything goes well, you will see a confirmation on your screen to provision. Type yes to proceed, and your resource will be live in a few minutes.

```
terraform apply
```

```
Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.
```

```
Enter a value: yes
```

```
aws_instance.example: Creating...  
aws_instance.example: Still creating... [10s elapsed]  
aws_instance.example: Still creating... [20s elapsed]  
aws_instance.example: Still creating... [30s elapsed]  
aws_instance.example: Still creating... [40s elapsed]  
aws_instance.example: Still creating... [50s elapsed]  
aws_instance.example: Still creating... [1m0s elapsed]  
aws_instance.example: Still creating... [1m10s elapsed]  
aws_instance.example: Still creating... [1m20s elapsed]  
aws_instance.example: Creation complete after 1m22s [id=i-0c56b6d7]
```

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.  
ubuntu@ip-172-31-94-180:~/terraform-lab$ █
```

Step 7) You can check your new AWS EC2 instance created in your AWS console. To delete all the resources created by Terraform, use the below command to clear all the things.

```
terraform destroy
```

Conclusion

Terraform is one of the most used Infrastructures as a Code tool by IT companies. In this blog, we created an AWS EC2 instance with Terraform using declarative syntax on our system. Using similar steps, we can create, manage and destroy any resource in AWS Infrastructure with the help of Terraform.

Terraform vs. Ansible: Key Differences and Comparison of Tools

Ansible vs Terraform battle continues to escalate as the DevOps environment focuses more on automation and orchestration. These two tools help in automating configurations and deploying infrastructure. Terraform offers to deploy Infrastructure as a Code, helps in readability and lift and shift deployments. Ansible is a configuration management tool for automating system configuration and management.

Terraform vs. Ansible

Terraform is a tool designed to help with the provisioning and deprovisioning of cloud infrastructure using an infrastructure as code approach. It is highly specialized for this purpose. On the other hand, Ansible is a more general tool that can be used for automation across various

domains. Both Terraform and ansible have strong open-source communities and commercial products that are well supported.

To explain in short, **Terraform is a tool used for creating and managing IT infrastructure. Ansible automates provisioning, deployment, and other IT processes.**

Similarities between Terraform and Ansible

Given the features of both technologies, Terraform and Ansible appear to be extremely similar tools at a high level.

- They are both capable of setting up the new cloud infrastructure and equipping it with the necessary application components.
- On the freshly formed virtual machine, remote commands can be carried out by both Terraform and Ansible. This indicates that neither tool requires an agent. Agent deployment on the computers is not necessary for operational reasons.
- Terraform builds infrastructure utilizing the APIs of cloud providers, and SSH is used for simple configuration operations. The same is true of Ansible; all necessary configuration activities are carried out over SSH. Both tools are masterless since the “state” information for neither requires a separate piece of infrastructure to manage.

Difference between Terraform and Ansible Provisioning (Terraform vs. Ansible)

Let's see how Terraform vs. Ansible battle differentiates from each other:

Terraform

Terraform is a provisioning tool.

It follows a declarative Infrastructure as a Code approach.
It is the best fit for orchestrating cloud services and setup
cloud infrastructure from scratch.

Terraform does not support bare metal provisioning by
default.

It does not provide better support in terms of packaging and
templating.

It highly depends on lifecycle or state management.

Ansible

Ansible is a configuration management tool.

It follows both declarative & procedural approaches.
It is mainly used for configuring servers with the
required software and updating already configured resources.

Ansible supports the provisioning of bare metal servers.

It provides full support for packaging and templating.

It does not have lifecycle management at all.

1. Orchestration vs. Configuration Management

Terraform and Ansible have so many similarities and differences at the same time. The difference comes when we look at two significant concepts of DevOps: Orchestration and configuration management.

Configuration management tools solve the issues locally rather than replacing the system entirely. Ansible helps to configure each action and instrument and ensures smooth functioning without any damage or error. In addition, Ansible comes up with hybrid capabilities to perform both orchestration and replace infrastructure.

Orchestration tools ensure that an environment is in its desired state continuously. Terraform is explicitly designed to store the state of the domain. Whenever there is any glitch in the system, terraform automatically restores and computes the entire process in the system after reloading. It is the best fit in situations where a constant and invariable state is needed. **Terraform Apply** helps to resolve all anomalies effectively.

Let's have a look at the Procedural and Declarative nature of Terraform and Ansible.

2. Declarative vs. Procedural

There are two main categories of DevOps tools: Procedural vs. Declarative. These two categories tell the action of tools.

Terraform follows the **declarative approach**, ensuring that if your defined environment suffers changes, it rectifies those changes. This tool attempts to reach the desired end state described by the *sysadmin*. Puppet also follows the declarative approach. With terraform, we can automatically describe the desired state and figure out how to move from one state to the next.

Ansible is of hybrid nature. It follows both declarative and **procedural style** configuration. It performs ad-hoc commands to implement procedural-style configurations. Please read the [documentation](#) of Ansible very carefully to get in-depth knowledge of its behavior. It's important to know whether you need to add or subtract resources to get the desired result or need to indicate the resources required explicitly.

3. Mutable vs. Immutable

A workflow for application deployment involves providing the infrastructure, installing the correct version of the source code, and installing any dependencies.

The infrastructure that serves as the foundation for later versions of apps and services has a property known as mutability. Either existing infrastructure is used for deployment, or we can create an entirely new set of infrastructure for it.

It depends on the deployment procedures whether the infrastructure is mutable or immutable. It is said to as malleable when subsequent versions of apps are released on the same infrastructure. However, it is said to as immutable if the deployment takes place during releases on entirely new infrastructure.

Although mutability appears convenient, there is a higher chance of failure. The previous version must first be uninstalled before the desired version can be installed when application configurations are applied again on the same infrastructure. Additional steps increase the likelihood of failure. This can lead to inconsistent setups and unpredictable behavior across a fleet of servers.

Instead, if we concentrate on minimizing these steps by skipping the uninstalling process and carrying out the installation on fresh infrastructure resources, we will have the opportunity to test the new deployment and roll it back in case it doesn't work. This approach to treating infrastructure as immutable gives administrators more control over making changes.

4. State Management

The full lifecycle of the resources under Terraform's administration is managed. It keeps up the state files' mapping of infrastructure resources to the most recent configuration. State management is crucial to Terraform's operation.

States are used to monitor configuration changes and provide the same. Additionally, it is possible to import preexisting resources managed by Terraform by bringing in state files from the infrastructure of the real world.

It is possible to query the Terraform state files at any moment to learn about the infrastructure components and their available characteristics.

Ansible, in contrast, does not provide any form of lifecycle management. Any changes made to the configuration are automatically implemented on the target resource because Ansible focuses on configuration management and assumes immutable infrastructure by default.

Terraform vs Ansible Provisioning



Terraform deals with **infrastructure automation**. Its current declarative model lacks some features which arise complexity. Using Terraform, the elements of required environments are separately described, including their relationships. It assesses the model, creates a plan based on dependencies, and gives optimized commands to Infrastructure as a Service. If there is no change in the environment or strategy, repeated runs will do nothing. If there is any **update** in the plan or environment, it will **synchronize** the cloud infrastructure.

Ansible follows a **procedural approach**. Various users create playbooks that are evaluated through top to bottom approach and executed in sequence. **Playbooks** are responsible for the configuration of network devices that contributes towards a procedural approach. Of course, Ansible provisions the cloud infrastructure as well. But its procedural approach limits it to large infrastructure deployments.

How does Terraform work?

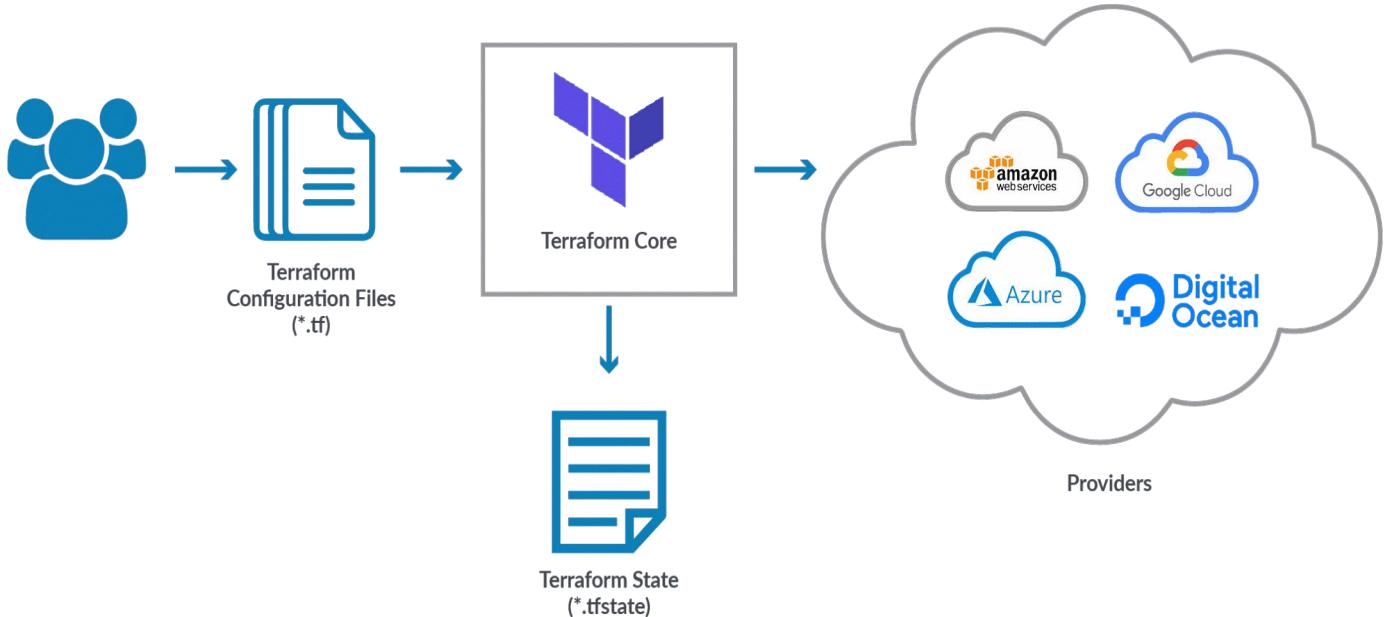
There are two main working components of terraform.

- **Terraform Core**
- **Providers**

Terraform is of **declarative nature**. It directly describes the end state of the system without defining the steps to reach there. It works at a high level of abstraction to describe what services and resources should be created and defined.

Terraform core takes two input sources to do its job. The first input source is a **terraform configuration** that is configured by its users. Users define what needs to be provisioned and created. The second input source is a state that holds information about the infrastructure.

So terraform core takes the input and figures out various plans for what steps to follow to get the desired output.



The second principal component is **providers**, such as cloud providers like [AWS](#), [GCP](#), [Azure](#), or other Infrastructure as service platforms. It helps to create infrastructure on different levels. Let's take an example where users create an AWS infrastructure, deploy Kubernetes on top of it, and then create services inside the cluster of Kubernetes. Terraform has multiple providers for various technologies; users can access resources from these providers through terraform. This is the basic working terminology of terraform that helps to provision and cover the complete application set up from infrastructure to fully developed application.

Check out: [Terraform cheat sheet](#).

Features of Terraform

As we have discussed the working of Terraform, now we will look at the features of Terraform.

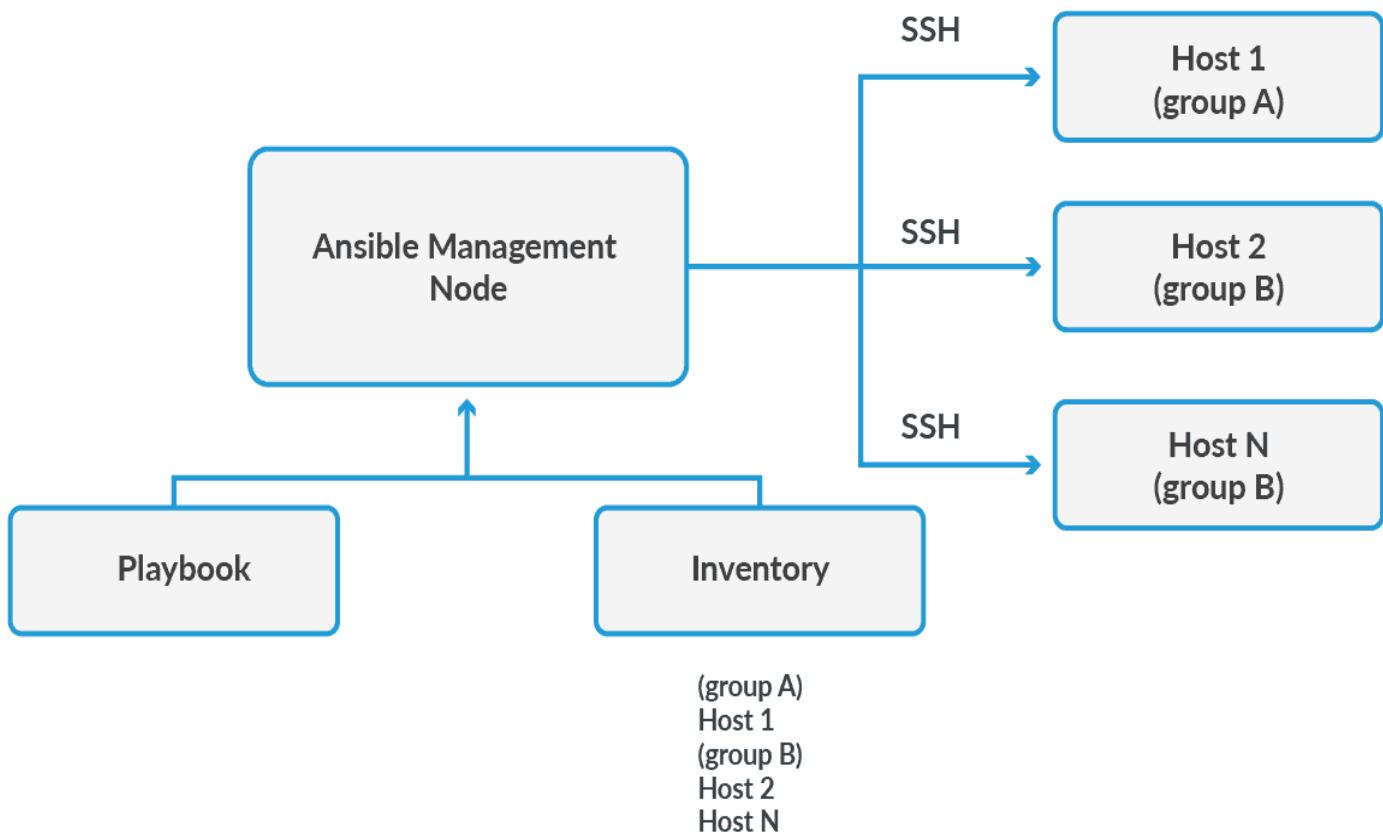
- Terraform follows a **declarative approach** which makes deployments fast and easy.
- It is a convenient tool to display the resulting model in a **graphical form**.
- Terraform also manages **external service providers** such as cloud networks and in-house solutions.
- It is one of the rare tools to offer **building infrastructure** from scratch, whether public, private or multi-cloud.
- It helps **manage parallel environments**, making it a good choice for testing, validating bug fixes, and formal acceptance.

- Modular code helps in achieving **consistency, reusability, and collaboration**.
- Terraform can **manage multiple clouds** to increase fault tolerance.

Also Check: Our blog post on [Terraform Workflow](#). Click here

How does Ansible work?

Ansible is agentless and doesn't run on target nodes. It makes **connections using SSH** or other authentication methods. It installs various **Python modules** on the target using JSON. These modules are simple instructions that run on the target. These modules are executed and removed once their job is done. This strategy ensures that there is no misuse of resources on target. Python is mandatory to be installed on both the controlling and the target nodes.



Ansible **management node** acts as a controlling node that controls the entire execution of the playbook. This node is the place to run the installations. There is an **inventory file** that provides the host list where the modules need to be run. The management node makes SSH connections to execute the modules on the host machine and installs the product. Modules are removed once they are installed in the system. This is the simple working process of Ansible.

Let's have a look at the features of ansible.

Features of Ansible

Now we will discuss various features Ansible provides to benefit its users.

- Ansible is used for **configuration management** and follows a procedural approach.
- Ansible deals with **infrastructure platforms** such as bare metal, cloud networks, and virtualized devices like hypervisors.

- Ansible follows **idempotent behavior** that makes it to place node in the same state every time.
- It uses **Infrastructure as a Code system configuration** across the infrastructure.
- It offers **rapid and easy deployment** of multi-tier apps with being agentless.
- If the code is **interrupted**, it allows entering the **code again** without any conflicts with other invocations.

Check out: [Ansible Configuration Management Tools](#)

Which one to choose: Terraform or Ansible?



Terraform vs. Ansible: Every tool has its unique characteristics and limitations. Let's check out which one to go with.

Terraform comes with good **scheduling capabilities** and is **very user-friendly**. It integrates with docker well, as docker handles the configuration management slightly better than Terraform. But there is no clear evidence of how the target devices are brought to their final state, and sometimes, the final configuration is unnecessary.

Ansible comes with **better security** and **ACL functionality**. It is considered a mature tool because it adjusts comfortably with **traditional automation frameworks**. It offers simple operations and helps to code quickly. But, on the other hand, it is not good at services like logical dependencies, orchestration services, and interconnected applications.

You can now choose between these two, according to the requirement of the situation and the job. For example, if the containerized solution is used to provision software within the cloud, then Terraform is preferable. On the other hand, if you want to gain reasonable control of your devices and find other ways to deploy underlying services, Ansible is more suitable. These tools will provide more comprehensive solutions in the future.

Conclusion

It is essential to know which tool is used for which job among Terraform vs. Ansible. Terraform is mainly known for provisioning infrastructure across various clouds. It supports more than 200 providers and a great tool to manage cloud services below the server. In comparison, Ansible is optimized to perform both provisioning and configuration management. Therefore, we can say that

both Terraform and Ansible can work hand in hand as **standalone tools** or **work together** but always pick up the right tool as per the job requirement.

Terraform Providers Overview

Terraform Providers: Terraform is one of the most popular tools used by DevOps teams to automate infrastructure tasks. It is used to provision and manage any cloud, infrastructure, or service.

Terraform officially supports around 130 providers. Its community-supported providers' page lists another 160. Some of those providers expose just a few resources, but others, such as AWS, OCI, or Azure, have hundreds of them.

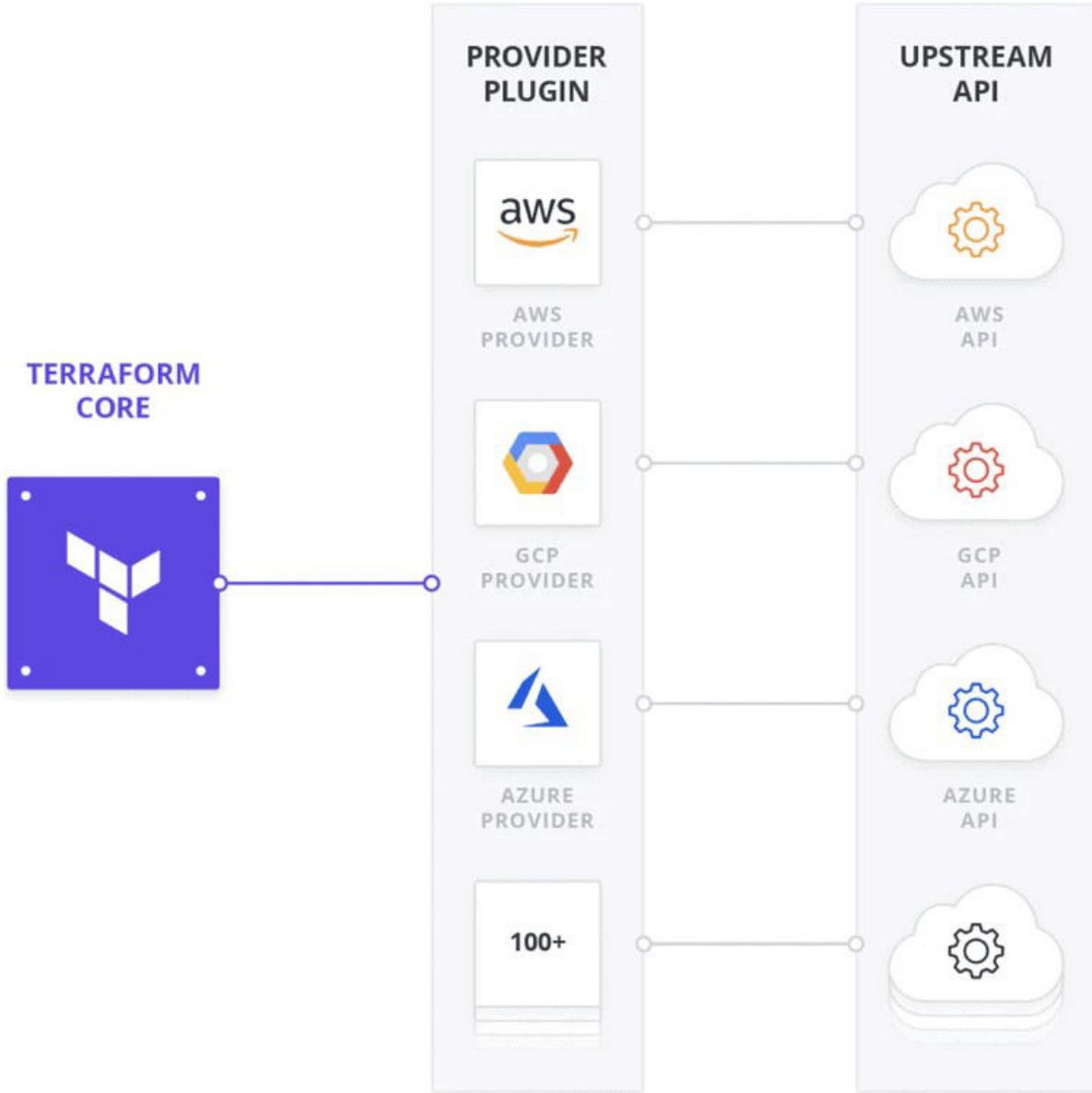
In this blog post, we cover a basic introduction of terraform providers and some major terraform cloud providers such as AWS, Azure, Google, and OCI.

A large percentage of Terraform users provision their infrastructure on the major cloud providers such as AWS, Azure, OCI, and others. To know more about various other terraform providers check [here](#).

Let's understand the basic terminologies often used in Terraform:

- **Terraform** provisions, updates, and destroys infrastructure resources such as physical machines, VMs, network switches, containers, and more.
- **Configurations** are code written for Terraform, using the human-readable HashiCorp Configuration Language (HCL) to describe the desired state of infrastructure resources.
- **Providers** are the plugins that Terraform uses to manage those resources. Every supported service or infrastructure platform has a provider that defines which resources are available and performs API calls to manage those resources.
- **Modules** are reusable Terraform configurations that can be called and configured by other configurations. Most modules manage a few closely related resources from a single provider.
- **The Terraform Registry** makes it easy to use any provider or module. To use a provider or module from this registry, just add it to your configuration; when you run `terraform init`, Terraform will automatically download everything it needs.

A provider is responsible for understanding API interactions and exposing resources. It interacts with the various APIs required to create, update, and delete various resources. Terraform configurations must declare which providers they require so that Terraform can install and use them.



Also check: Types of [Terraform Variables](#)

Terraform AWS Provider

Terraform can provision infrastructure across public cloud providers such as Amazon Web Services (AWS), Azure, Google Cloud, and DigitalOcean, as well as private cloud and virtualization platforms such as OpenStack and VMWare.

AWS is a good choice for learning Terraform because of the following:

- AWS is the most popular cloud infrastructure provider, by far. It has a 45% share in the cloud infrastructure market, which is more than the next [three biggest competitors](#) (Microsoft, Google, and IBM) combined.
- AWS provides a huge range of reliable and scalable cloud hosting services, including Amazon Elastic Compute Cloud (Amazon EC2), which you can use to deploy virtual servers; Auto Scaling Groups (ASGs), which make it easier to manage a cluster of virtual servers; and Elastic Load Balancers (ELBs), which you can use to distribute traffic across the cluster of virtual servers.

- AWS offers a generous Free Tier for the first year that should allow you to run all of these examples for free. If you already used up your free tier credits, the examples in this book should still cost you no more than a few dollars.

Learn more about how to [AWS Free Tier Account](#) to avail the free tier services.

The two most popular options for deploying infrastructure to AWS are [CloudFormation](#), a service native to AWS, and [Terraform](#), an open-source offering from HashiCorp. Most of the AWS resources can be provisioned with Terraform as well and is often faster than CloudFormation when it comes to supporting new AWS features. On top of that, Terraform supports other cloud providers as well as 3rd party services.

Also read: Step by step guide on [Terraform Certification](#)

Azure Terraform Resource Provider

Azure Resource Providers for HashiCorp [Terraform](#) enables Azure customers using Azure Resource Manager (ARM) to provision and manage their resources with Terraform Providers as if they were native Azure Resource Providers.

Below are some of the core infrastructure services supported by Azure Resource Provider in Terraform:

- Virtual machines
- Storage accounts
- Networking interfaces
- [Azure Container Instances](#)
- [Azure Container Service](#)
- [Managed Disks](#)
- [Virtual Machine Scale Sets](#)

The ARM Resource Provider leverages HashiCorp Terraform to provide third-party services to ARM users directly via ARM. Some of these third-party services supported are listed below:

- [Kubernetes](#)
- [Datadog](#)
- [Cloudflare](#)

Terraform is built into [Azure Cloud Shell](#) and cloud shell automatically authenticates your default Azure CLI subscription to deploy resources through the Terraform Azure modules.

To know more about Azure provider for Terraform, click [here](#).

Also Check: Our blog post on [Terraform Commands Cheat Sheet](#). Click here

Oracle Cloud Infrastructure Terraform Provider

Oracle Cloud Infrastructure is an official provider of Hashicorp Terraform supporting infrastructure-as-code for oracle cloud customers.

The provider is compatible with Terraform 0.10.1 and later. Following are some of the main resources supported by the Terraform provider:

- [Block Volumes](#)
- [Compute](#)
- [Container Engine](#)
- [Database](#)
- [File Storage](#)
- [Identity and Access Management \(IAM\)](#)
- [Load Balancing](#)
- [Networking](#)
- [Object Storage](#)

A detailed list of supported resources and more information about how to get started is available on the [HashiCorp website](#).

Oracle also provides **Resource Manager**, a fully managed service to operate Terraform. Resource Manager integrates with Oracle Cloud Infrastructure Identity and Access Management (IAM), so you can define granular permissions for Terraform operations. It also provides state locking, giving users the ability to share state, and lets teams collaborate effectively on their Terraform deployments. Most of all, it makes operating Terraform easier and more reliable.

To know more about Resource Manager, check [here](#).

Oracle had announced two features to help you bring your existing infrastructure to Terraform and Resource Manager:

- **Terraform Resource Discovery** helps you move from a console- or SDK-managed infrastructure to an infrastructure managed by Terraform.
- **State File Import** helps customers who have a Terraform-managed infrastructure move to a Resource Manager-managed infrastructure.

To know more about Terraform Resource Discovery, check [here](#).

Now that we got an overview of what a provider is and services provided by some major providers, let's see how we can use one in our terraform configuration files.

Also Read: Our blog post on [Hashicorp Terraform](#). Click here

Google Cloud (GCP) Terraform Provider

The Google Cloud Terraform Provider is used to configure your [Google Cloud Platform](#) infrastructure. It is collaboratively maintained by the Google Terraform Team at Google and the Terraform team at HashiCorp.

It looks over the lifecycle management of GCP resources, including Compute Engine, Cloud Storage, Cloud SDK, Cloud SQL, GKE, BigQuery, Cloud Functions, and more.

Below is a general provider configuration snippet:

```
provider "google" {
  project      = "my-project-id"
  region       = "us-central1"
}
```

In order to get started with Google Terraform Provider, you need to have a Google Cloud Free Trial Account with billing enabled and terraform installed in your project. Follow the steps from this guide to create your [Google Cloud Free Trial Account](#) and create a project.

Provider Block

Terraform configurations must declare which providers they require so that Terraform can install and use them. Providers are executable plugins that contain the code necessary to interact with the API of the service it was written for.

A provider is defined by a **provider** block, the actual arguments in a provider block vary depending on the provider, but all providers support the meta-arguments of version and alias.

The below image shows the provider block format across different providers.

```
provider azurerm {
  version = "1.41.0"
  tenant_id = var.tenant_id
  subscription = var.subscription_id
}

# Configure the AWS Provider
provider "aws" {
  version = "~> 3.0"
  region  = "us-east-1"
}

# Configure the Oracle Cloud Infrastructure provider with an API Key
provider "oci" {
  tenancy_ocid = "${var.tenancy_ocid}"
  user_ocid = "${var.user_ocid}"
  fingerprint = "${var.fingerprint}"
  private_key_path = "${var.private_key_path}"
  region = "${var.region}"
}
```

Check Out: How to [Install Terraform](#) on Mac, Windows & Ubuntu. Click here

Provider Workflow

Terraform finds and installs providers when initializing a working directory. It can automatically download providers from a Terraform registry, or load them from a local mirror or cache.

Hashicorp distributed providers are available for download automatically during Terraform initialization, while third-party providers must be placed in a local plug-ins directory located at either %APPDATA%\terraform.d\plugins for Windows or ~/.terraform.d/plugins for other operating systems.



The flow of steps performed is explained below:

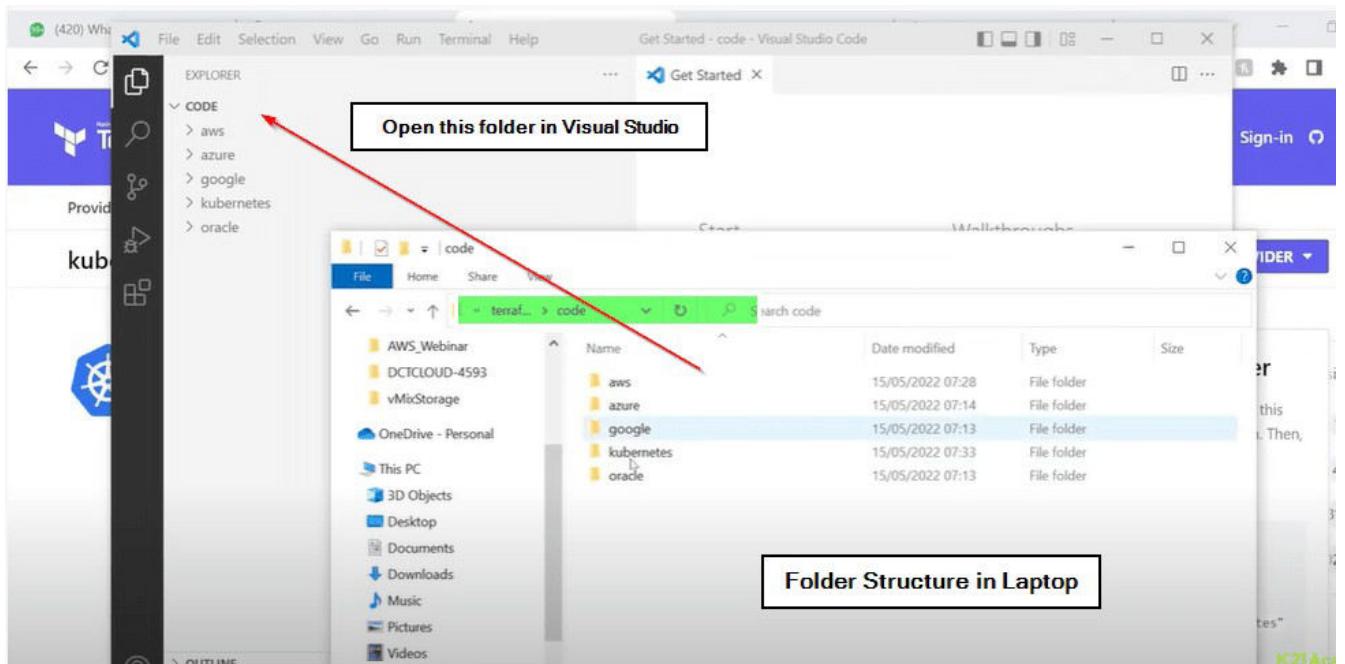
1. Initialize the Terraform configuration, i.e. run `terraform init` command.
2. It looks for provider being used, and download the provider plug-ins, if not found.
3. Retrieves the provider plug-ins
4. Stores them in a `.terraform` subdirectory
5. Check the default version or specified version

How To Use Providers

You can follow some simple steps to use your choice of provider be it AWS, Azure, Google Cloud, or Oracle in any IDE of your choice (here we are using VSCode- Visual Studio Code).

On your laptop, you can create a folder names Terraform and inside this folder create sub-folders like AWS, Azure, Google, etc.

Then open this complete folder in VSCode and also install the HashiCorp Terraform extension.



Now suppose you have to use the AWS provider, then in VSCode inside the AWS folder create a new file as the `provider.tf` and copy the code from the official [AWS provider](#) site in that file. And once done you need to initialize the terraform by running `terraform init` command in the terminal.

Providers / hashicorp / aws / Version 4.17.0 Latest Version

aws

aws

Official by HashiCorp

Public Cloud

Lifecycle management of AWS resources, including EC2, Lambda, EKS, ECS, VPC, S3, RDS, DynamoDB, and more. This provider is maintained internally by the HashiCorp AWS Provider team.

VERSION PUBLISHED SOURCE CODE

4.17.0 5 hours ago hashicorp/terraform-provider-aws

How to use this provider

To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.17.0"
    }
  }
}

provider "aws" {
  # Configuration options
}
```

Terraform 0.14+

```
provider "aws" {
  # Configuration options
}
```

EXPLORER

CODE

aws provider.tf

Create a file inside the folder

Get Started provider.tf

```
aws > provider.tf > provider "aws" ②
1  terraform {
2    required_providers {
3      aws = {
4        source = "hashicorp/aws"
5        version = "4.14.0"
6      }
7    }
8  }
9
10 provider "aws" {
11   # Configuration options
12 }
```

PROBLEMS OUTPUT TERMINAL ③ Open Terminal

Directory: C:\terraform\code\aws

Mode	LastWriteTime	Length	Name
-a---	18/05/2022 17:57	171	provider.tf

PS C:\terraform\code\aws> `terraform init` ④

Move to aws folder and write the command

K21Academy

Download Terraform

<https://www.terraform.io/downloads.html>

Terraform CLI Documentation

<https://www.terraform.io/docs/cli/index.html>

Course: Using Terraform to Manage Applications and Infrastructure

<https://bit.ly/3Bkr9nT>