Learning Route

===============

Linux PDF by ratnakar

LDAP

samba CIFS

Stephan maarek AWS both courses

UNIX AIX solaris

Pacemaker HA

red hat sattelite

openshift RED hat

Devops course all

CICD: GIT Jenkins, Docker, Kubernetes, Terraform/Ansible

KVM, OpenStack, Linux, Gitlab.

veritas cluster

VMware course

Vrealize

Kubernetes course

========================================================================

CentOS fix for mirrors

sed -i s/mirror.centos.org/vault.centos.org/g /etc/yum.repos.d/CentOS-\*.repo

sed -i s/^#.\*baseurl=http/baseurl=http/g /etc/yum.repos.d/CentOS-\*.repo

sed -i s/^mirrorlist=http/#mirrorlist=http/g /etc/yum.repos.d/CentOS-\*.repo

Here are the step-by-step instructions for setting up a Pacemaker cluster:

A Pacemaker cluster is a high-availability cluster used to manage resources and ensure that services are always available, even in the event of node or resource failures. Below is a detailed guide to setting up and managing a Pacemaker cluster.

---

1. Prerequisites

Before starting, ensure the following:

Two or more servers for cluster nodes.

A Linux distribution (e.g., RHEL, CentOS, Debian, or Ubuntu) installed on all nodes.

Proper hostname and static IP address configured for each node.

Passwordless SSH access between nodes.

NTP or Chrony configured for time synchronization.

Corosync and Pacemaker installed.

---

2. Installing Pacemaker and Corosync

On all cluster nodes, run:

sudo apt install pacemaker corosync pcs -y # For Debian/Ubuntu

sudo yum install pacemaker corosync pcs -y # For RHEL/CentOS

Enable and start the pcs daemon:

sudo systemctl enable pcsd

sudo systemctl start pcsd

---

3. Cluster Setup and Configuration

Step 1: Configure Cluster Nodes

1. Set a common password for hacluster user (used by pcs):

echo "password" | passwd hacluster

2. Authenticate nodes:

sudo pcs cluster auth node1 node2

Replace node1 and node2 with your node hostnames or IPs.

---

Step 2: Create the Cluster

1. Create the cluster:

sudo pcs cluster setup --name my\_cluster node1 node2

2. Start the cluster:

sudo pcs cluster start --all

3. Check cluster status:

sudo pcs status

---

Step 3: Configure Fencing (STONITH)

Fencing ensures data integrity by powering off an unresponsive node.

1. Identify your fencing mechanism (e.g., IPMI, iLO, AWS, etc.).

2. Configure the fencing device:

sudo pcs stonith create stonith-device fence\_xvm pcmk\_host\_list="node1 node2"

3. Enable fencing:

sudo pcs property set stonith-enabled=true

---

Step 4: Configure Resources

Pacemaker manages resources such as IP addresses, services, and filesystems. For example, to manage an Apache service:

1. Add a resource:

sudo pcs resource create apache ocf:heartbeat:apache configfile=/etc/httpd/conf/httpd.conf op monitor interval=30s

2. Set resource constraints:

sudo pcs constraint colocation add apache with stonith-device INFINITY

sudo pcs constraint order stonith-device then apache

---

Step 5: Test Failover

1. Simulate a failure:

sudo systemctl stop pacemaker

2. Observe that the resource fails over to another node:

sudo pcs status

---

Step 6: Fine-Tune Cluster Properties

Adjust properties to suit your use case:

sudo pcs property set no-quorum-policy=ignore

sudo pcs property set default-resource-stickiness=100

---

4. Monitoring and Maintenance

Check Cluster Logs:

Logs are in /var/log/pacemaker.log and /var/log/corosync.log.

Add a Node to the Cluster:

sudo pcs cluster node add newnode

Remove a Node from the Cluster:

sudo pcs cluster node remove oldnode

---

5. Advanced Configurations

DRBD for Shared Storage:

Configure DRBD for high-availability storage replication between nodes.

Cluster Web GUI:

Install pcs-gui for a graphical interface to manage the cluster.

---

This setup provides a basic yet robust foundation for a Pacemaker cluster. For more advanced use cases, refer to the Pacemaker documentation.

Step 1: Plan the Cluster

1. Determine the number of nodes

2. Choose the cluster type (Active/Passive, Active/Active, N+1)

3. Identify resources to manage (e.g., Apache, MySQL)

4. Plan the network infrastructure

Step 2: Install Pacemaker and Corosync

1. Install Pacemaker and Corosync on each node:

- yum install pacemaker corosync (RHEL/CentOS)

- apt-get install pacemaker corosync (Ubuntu/Debian)

2. Verify installation:

- rpm -qa | grep pacemaker (RHEL/CentOS)

- dpkg -l | grep pacemaker (Ubuntu/Debian)

Step 3: Configure Corosync

1. Create the Corosync configuration file:

- /etc/corosync/corosync.conf

2. Define the cluster name, node names, and network settings:

- cluster\_name: mycluster

- `node {

- name: node1

- nodeid: 1

- ``

- }

3. Save and close the file

Step 4: Start Corosync

1. Start Corosync on each node:

- systemctl start corosync (RHEL/CentOS)

- service corosync start (Ubuntu/Debian)

2. Verify Corosync status:

- corosync-cmapctl | grep membership

Step 5: Configure Pacemaker

1. Create the Pacemaker configuration file:

- /etc/pacemaker/pacemaker.conf

2. Define the cluster properties:

- cluster-property no-quorum-policy="ignore"

- cluster-property stonith-enabled="false"

3. Save and close the file

Step 6: Define Resources

1. Create resource definitions:

- crm configure primitive apache ocf:heartbeat:apache params configfile="/etc/httpd/conf/httpd.conf"

- crm configure primitive mysql ocf:heartbeat:mysql params database="mydb" user="myuser" password="mypass"

2. Verify resource definitions:

- crm configure show

Step 7: Define Constraints

1. Define constraints for resource ordering and colocation:

- crm configure colocation apache mysql INFINITY

- crm configure order apache mysql

2. Verify constraint definitions:

- crm configure show

Step 8: Start Pacemaker

1. Start Pacemaker on each node:

- systemctl start pacemaker (RHEL/CentOS)

- service pacemaker start (Ubuntu/Debian)

2. Verify Pacemaker status:

- crm status

Step 9: Test the Cluster

1. Test resource failover:

- crm resource failover apache

- crm resource failover mysql

2. Verify cluster status:

- crm status

Step 10: Monitor and Maintain

1. Monitor cluster status:

- crm status

2. Perform regular backups:

- crm configure backup

3. Update Pacemaker and Corosync:

- yum update pacemaker corosync (RHEL/CentOS)

- apt-get update pacemaker corosync (Ubuntu/Debian)

======================================================================================================================

Components Red hat sattelite

1. Foreman: Provisioning and configuration management

2. Katello: Content management and subscription management

3. Candlepin: Subscription management and entitlement

4. Pulp: Content management and repository management

======================================================================================================================

GPG Key:

A GPG (GNU Privacy Guard) key is a digital key used for encrypting, decrypting, and signing data.

What is a GPG Key?

A GPG key is a pair of keys:

1. Public Key: used for encrypting data

2. Private Key: used for decrypting data

GPG Key Features:

1. Encryption: secure data transmission

2. Decryption: access encrypted data

3. Digital Signatures: authenticate data integrity

4. Key Management: manage public and private keys

======================================================================================================================

tcpdump -i eth0 capturing packets

======================================================================================================================

RAID DP (NetApp-specific): uses two parity blocks and one spare block

======================================================================================================================

SAR stands for System Activity

messages are stored in /var/log/sa/sa1,

/var/log/sa/sa2, /var/log/sa/sa3

Config: /etc/sysconfig/sysstat

======================================================================================================================

For CPU info

lscpu

nproc

dmidecode

dmidecode -t bios

======================================================================================================================

firewall-cmd --permanent --add-service=ftp

firewall-cmd --permanent --add-port=1234/tcp

======================================================================================================================

NFSv4 Components:

1. Server: The NFSv4 server is responsible for sharing files and directories with clients.

2. Client: The NFSv4 client is responsible for accessing shared files and directories on the server.

3. Portmap Service: The portmap service is responsible for mapping NFS requests to the correct TCP/IP port.

4. rpc.mountd Service: The rpc.mountd service is responsible for managing NFS mounts and exports.

5. rpc.nfsd Service: The rpc.nfsd service is responsible for managing NFS daemons.

nfsiostat -s to check nfs usage

no\_root\_squash:

The no\_root\_squash option is a parameter used in the NFS (Network File System) protocol to control the behavior of the root user on a NFS client.

======================================================================================================================

SMB (Server Message Block) and CIFS (Common Internet File System)

Serverside

------------

yum install samba samba-client samba-common

firewall-cmd --permanent --zone=public --add-service=samba

firewall-cmd --reload

mkdir -p /samba/apps

chcon -t samba\_share\_t /samba/apps/

systemctl start smb nmb

vi /etc/samba/smb.conf

[global]

workgroup = SAMBA

netbios name = centos

security = user

map to guest = bad user

dns proxy = no

interface = eth0 vmnet1

[Apps]

comment = Shared Dir

path = /samba/apps

browsable = yes

writable = yes

guest = ok

guest only = yes

read only = no

testparm to test the changes

systemctl restart smb nmb

groupadd sambashare

add user testuser to sambashare

mkdir sambashare

chmod 2770 samba\_secure/

chcon -t samba\_share\_t /samba\_secure/

[secure]

path = /samba\_secure/

read only = No

valid users = @sambagroup

smbpasswd -a testuser add user to smb service

smbpasswd -e testuser enable user to smb service

Client side server

----------------------

yum install cifs-utils samba-client

mkdir /mnt/samba/apps

mkdir -p /mnt/samba/apps

mount -t cifs //192.168.222.133/apps /mnt/samba/apps/

df -hT

[root@localhost ~]# df -hT /mnt/samba/apps

Filesystem Type Size Used Avail Use% Mounted on

//192.168.222.133/apps cifs 19G 2.3G 17G 12% /mnt/samba/apps

======================================================================================================================

FAAS function as service

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS) that allows developers to run code without provisioning or managing servers. Lambda automatically handles the infrastructure, scaling, and high availability of the application, allowing you to focus purely on your code.

======================================================================================================================

AWS Elastic Beanstalk is a fully managed service from Amazon Web Services that simplifies deploying, managing, and scaling web applications and services. It abstracts much of the infrastructure management, allowing developers to focus more on coding rather than handling servers and scaling concerns.

======================================================================================================================

Elastic beanstalk and lambda are very different though some of the features may look similar. At high level, elastic beanstalk deploys a long running application whereas lambda deploys short running code function

Lambda can at maximum run for 15 minutes, whereas EB can run continuously. Generally, we deploy websites/apps on EB whereas lambda are generally used for triggered functionality like processing image when image gets uploaded to S3.

Lambda can only handle one request at a time whereas number of concurrent requests EB can handle depends on your underlying infrastructure. So, if you are having say 100 requests, 100 lambdas will be created whereas these 100 requests can be handled by one underlying EC2 instance in EB

Lambda is serverless (underlying infra is entirely abstracted from developer). Whereas EB is automation over infra provisioning. You can still see your EC2 instances, load balancer, auto scaling group etc. in your AWS console. You can even ssh/rdp to your instance and change running services. AWS EB allows you also to have your custom AMIs.

Lambda is having issue of cold starts as in lambda, infra needs to be provisioned on demand by AWS, whereas in EB, you generally have EC2 instances already provisioned to handle your requests.

======================================================================================================================

Key **Key Differences:**

| **Feature** | **Worker Environment** | **Web Server Environment** |
| --- | --- | --- |
| **Primary Purpose** | Handling background tasks, queues, and jobs. | Handling client HTTP requests and responses. |
| **User Interaction** | No direct interaction with users. | Direct interaction with users (e.g., rendering web pages). |
| **Task Type** | Asynchronous, background, time-consuming tasks. | Synchronous, real-time tasks related to web request/response cycle. |
| **Scalability** | Can scale independently of the web server. | Often scales with load balancing, distributing incoming requests. |
| **Examples** | Background job queues (Celery, Sidekiq). | Web servers (Apache, Nginx, Node.js). |

======================================================================================================================

Amazon S3 (Simple Storage Service)

Type: Object Storage

Use Cases: Storing unstructured data like media files, backups, and log files.

Key Features:

Storage Classes:

Standard: High durability and availability for frequently accessed data.

Intelligent-Tiering: Automatically moves data to the most cost-effective access tier.

Glacier: Cost-effective storage for archival with retrieval options ranging from minutes to hours.

Scalability: Virtually unlimited storage capacity.

Data Security: Server-side encryption, versioning, and access control.

Examples: Hosting static websites, data lakes, and media delivery.

Amazon EBS (Elastic Block Store)

Type: Block Storage

Use Cases: Attaching persistent storage volumes to Amazon EC2 instances.

Key Features:

Performance Types:

General Purpose SSD (gp3/gp2): Balanced performance for most workloads.

Provisioned IOPS SSD (io2/io1): High-performance workloads like databases.

Magnetic Volumes (sc1/st1): Cost-effective storage for less frequently accessed data.

Snapshots: Enables backup and restoration of volumes.

Examples: Databases, transactional workloads, and boot volumes.

Amazon EFS (Elastic File System)

Type: File Storage

Use Cases: Sharing file storage across multiple EC2 instances or containers.

Key Features:

Fully managed, scalable file storage.

Supports NFS protocol, making it compatible with Linux-based systems.

Offers Standard and Infrequent Access storage tiers.

Examples: Content management, development environments, and analytics.

AWS Glacier

Type: Archival Storage

Use Cases: Long-term archival and regulatory compliance.

Key Features:

Ultra-low-cost storage.

Flexible retrieval options: Expedited, Standard, and Bulk retrieval modes.

High durability (11 nines).

Examples: Archiving financial records or medical imaging.

======================================================================================================================

Amazon VPC (Virtual Private Cloud)

Purpose: Enables users to create a logically isolated section of AWS for launching and managing resources in a customizable network.

Features:

Subnets (public and private).

Route tables, NAT gateways, and internet gateways for traffic routing.

Security groups and network ACLs for access control.

Use Cases:

Hosting secure applications.

Isolating workloads for compliance and security.

Amazon Route 53

Purpose: A scalable Domain Name System (DNS) service for routing end-user traffic to applications.

Features:

DNS management and health checks.

Traffic flow with latency-based, geolocation, or failover routing policies.

Use Cases:

Managing domain names for websites.

Ensuring low-latency user connections worldwide.

AWS Direct Connect

Purpose: Establishes a dedicated network connection between your on-premises environment and AWS.

Features:

Reduces latency compared to internet-based connections.

Ensures secure and reliable data transfer.

Use Cases:

Hybrid cloud architectures.

High-performance data transfer for large datasets.

======================================================================================================================

Amazon CloudFront is a fast, secure, and scalable Content Delivery Network (CDN) service provided by AWS. It is designed to deliver data, videos, applications, and APIs to users globally with low latency and high transfer speeds, leveraging AWS's robust infrastructure.

Uses edge location service

======================================================================================================================

Key Features of Amazon SNS

Topic-Based Messaging:

Allows publishers to send messages to a topic.

Subscribers (e.g., AWS Lambda, SQS, email, or SMS) receive these messages if subscribed to the topic.

Flexible Delivery:

Supports multiple protocols such as HTTP/HTTPS, email, SMS, SQS, and mobile push notifications.

High Availability and Scalability:

Automatically scales to handle millions of messages.

Message Filtering:

Subscribers can filter messages they want to receive based on message attributes.

Message Encryption:

Supports encryption at rest using AWS Key Management Service (KMS).

Dead-Letter Queues:

Captures undeliverable messages for debugging or analysis.

Mobile Push Notifications:

Direct integration with Apple Push Notification Service (APNs), Google Firebase Cloud Messaging (FCM), and others.

======================================================================================================================

Common Use Cases

Application Alerts:

Notify system administrators of critical application events.

Workflow Messaging:

Coordinate microservices using event-driven architectures.

Broadcast Notifications:

Send updates or promotions to customers via SMS or email.

Mobile Push Notifications:

Deliver app-specific updates to end users.

======================================================================================================================

Amazon CloudTrail is an AWS service that provides a detailed log of API and console actions performed on your AWS account. These logs help in monitoring, auditing, and ensuring compliance with security standards.

======================================================================================================================

Amazon CloudWatch

Metrics Monitoring:

Collects and tracks metrics from AWS resources like EC2, S3, RDS, and custom metrics from applications.

Provides pre-configured dashboards to visualize performance.

Alarms:

Alerts you when metrics exceed predefined thresholds.

Can trigger actions like scaling EC2 instances or sending notifications through SNS.

Logs:

Aggregates logs from services such as Lambda, ECS, and custom applications.

Enables searching, analyzing, and visualizing log data.

How to Set Up Amazon CloudWatch

Access CloudWatch:

Navigate to the AWS Management Console > CloudWatch.

Enable Metrics Collection:

Default AWS services automatically provide basic metrics (e.g., CPU utilization for EC2).

Use the AWS SDK or CloudWatch Agent for custom metrics.

Set Up Alarms:

In the Alarms section, define a metric and set thresholds.

Choose actions, such as notifying through SNS or invoking Lambda.

Configure Logs:

Use the CloudWatch Logs Agent or Fluent Bit to send log data.

Organize logs into log groups for analysis.

Create Dashboards:

Add widgets for metrics or logs.

Customize layout and content to suit operational needs.

Analyze with Insights:

Use Log Insights to run queries for specific patterns or performance issues.

Use Cases

Application Performance Monitoring:

Track latency, errors, and resource utilization.

Incident Detection and Response:

Receive alerts for system failures or anomalies.

Operational Optimization:

Identify cost-saving opportunities by analyzing resource utilization.

Compliance and Auditing:

Maintain a centralized view of logs for auditing purposes.

Pricing

Free Tier:

Basic monitoring for AWS resources.

Up to 10 metrics and 5 alarms are free.

Pay-As-You-Go:

Custom metrics: $0.30 per metric per month.

Logs: $0.50 per GB ingested.

Insights: $0.005 per query for logs and metrics.

======================================================================================================================

# AWS Default Metrics:

Automatically provided by AWS for supported services.

Examples:

EC2: CPU utilization, network I/O, disk read/write operations.

RDS: Database connections, read/write IOPS.

S3: Bucket size, number of objects, request latency.

Lambda: Invocations, duration, errors, throttles.

ALB/ELB: Request count, healthy/unhealthy hosts, latency.

Custom Metrics:

Metrics created by applications or systems outside of the default AWS services.

Examples:

Application-specific data like user logins, transaction counts, or business KPIs.

You can publish custom metrics using the AWS SDK, AWS CLI, or CloudWatch Agent.

======================================================================================================================

AWS CloudWatch Alarms

Amazon CloudWatch Alarms are a feature within Amazon CloudWatch that allow you to monitor metrics and take automated actions when specific conditions are met. These alarms are critical for ensuring system health, automating responses, and alerting when performance thresholds or anomalies occur.

======================================================================================================================

AWS CloudWatch Logs

Amazon CloudWatch Logs is a feature that collects and monitors log data from various AWS resources, applications, and on-premises systems. It provides capabilities for storing, searching, and analyzing log data to monitor system behavior, troubleshoot issues, and ensure operational excellence.

======================================================================================================================

AWS CloudFormation

AWS CloudFormation is an Infrastructure as Code (IaC) service that enables you to define and provision AWS infrastructure using code templates. By leveraging templates written in JSON or YAML, CloudFormation automates the creation and management of AWS resources, ensuring consistency and repeatability.

Create stack and drag drop services infra as code

Template-Based Provisioning:

Define infrastructure as code using templates.

Templates specify the resources and their configurations.

Automated Resource Management:

Deploy and manage AWS resources like EC2 instances, S3 buckets, and RDS databases.

Handles resource dependencies automatically.

======================================================================================================================

AWS Snapshot

AWS snapshots are a feature of Amazon Elastic Block Store (EBS) that allows you to create point-in-time backups of your EBS volumes. Snapshots are stored in Amazon S3 and can be used to restore EBS volumes to the state at the time the snapshot was created.

======================================================================================================================

Amazon Machine Image (AMI)

An Amazon Machine Image (AMI) is a pre-configured virtual machine template that contains all the necessary information to launch an instance in the AWS cloud. AMIs are critical for managing virtual servers (Amazon EC2 instances) in AWS.

======================================================================================================================

AWS Load Balancer

Amazon Web Services (AWS) offers a service called Elastic Load Balancing (ELB), which automatically distributes incoming application or network traffic across multiple targets such as EC2 instances, containers, IP addresses, and Lambda functions. ELBs are a critical component for building scalable and resilient applications in AWS.

Types of AWS Load Balancers

AWS provides four types of load balancers:

Application Load Balancer (ALB):

Operates at the application layer (Layer 7).

Best suited for HTTP and HTTPS traffic.

Features include:

Path-based routing (e.g., /api vs. /user).

Host-based routing (e.g., example.com vs. api.example.com).

Integration with AWS Web Application Firewall (WAF).

SSL termination for HTTPS traffic.

Use Cases: Web applications, microservices, and API services.

Network Load Balancer (NLB):

Operates at the transport layer (Layer 4).

Handles TCP, UDP, and TLS traffic.

Features include:

Static IP support.

High throughput and low latency.

Ability to handle millions of requests per second.

Use Cases: High-performance, real-time applications like gaming or financial systems.

Gateway Load Balancer (GWLB):

Operates at Layer 3 (Network layer).

Primarily used to deploy, scale, and manage virtual appliances such as firewalls, intrusion detection systems, and deep packet inspection systems.

Enables seamless traffic inspection and routing.

Classic Load Balancer (CLB):

Operates at both Layer 4 and Layer 7.

Legacy service; recommended to use ALB or NLB for new deployments.

Basic routing capabilities compared to ALB and NLB.

Key Features

Health Checks:

ELB performs health checks on targets and routes traffic only to healthy instances.

Automatic Scaling:

Automatically adjusts capacity to handle varying levels of traffic.

SSL Termination and Encryption:

Offloads SSL decryption to the load balancer.

Supports HTTPS to ensure secure communication.

Cross-Zone Load Balancing:

Distributes traffic evenly across all instances in different availability zones.

High Availability:

ELB is designed for fault tolerance and availability across multiple availability zones.

Integration with AWS Services:

Works seamlessly with Auto Scaling, EC2, ECS, EKS, and Lambda.

Go to load balancer > Create application load balancer > name "liveloadbalancer> Select internetfacing port 80 will be open for traffic> Avalailibity zones> Create new security group allow port 80 tcp > configure routing

Target group > liveauto1 is target group > create another target grpup liveauto2 > needs to point to autoscaling groups > Open autoscaling groups > Edit add target groups here

first create auto scaling groups and connect them with target groups

Create rule > http request will go to first load balancer

Image download request will go to second load balancer

======================================================================================================================

AWS Cloud Security is a combination of tools, policies, and best practices designed to protect the infrastructure, data, and applications running on the AWS cloud. Here's an overview of key features and strategies for securing workloads in AWS:

1. Shared Responsibility Model

AWS operates under a shared responsibility model, where:

AWS is responsible for the security of the cloud (physical infrastructure, managed services, etc.).

Customers are responsible for security in the cloud (data, applications, user access).

2. Key AWS Cloud Security Features

Identity and Access Management (IAM)

IAM Roles and Policies: Define who can access what resources with fine-grained permissions.

Multi-Factor Authentication (MFA): Adds an extra layer of authentication to secure root and IAM accounts.

Data Protection

Encryption: AWS supports data encryption in transit (SSL/TLS) and at rest (KMS).

Services like S3, RDS, and EBS offer integrated encryption options.

AWS Key Management Service (KMS): Manages encryption keys for applications.

Network Security

Amazon Virtual Private Cloud (VPC): Isolates your infrastructure within a private network.

Security Groups and Network ACLs: Act as virtual firewalls to control inbound and outbound traffic.

AWS Shield: Protects against DDoS attacks.

AWS WAF: Filters and blocks malicious web traffic.

Monitoring and Logging

AWS CloudTrail: Tracks all API activity in your account for auditing.

Amazon GuardDuty: Provides threat detection using machine learning.

AWS CloudWatch: Monitors application and system metrics for anomalies.

Application Security

AWS Secrets Manager: Safely stores and rotates secrets like API keys and passwords.

Amazon Inspector: Automates security assessments for applications.

AWS CodePipeline & CodeBuild: Integrate security scans into your CI/CD pipeline.

======================================================================================================================

AWS Identity and Access Management (IAM) is a powerful security tool that enables you to manage access to AWS resources. It ensures that only authorized users and services can access specific resources, while providing robust control and monitoring of permissions.

Key Concepts of AWS IAM:

Users:

IAM Users are individual identities created to represent people or services that need access to your AWS environment. Each user has credentials (like passwords and access keys) to authenticate and gain access to resources.

Users can be assigned permissions directly or through IAM groups.

Groups:

IAM Groups are collections of IAM users. You can apply the same permissions to multiple users by placing them in a group. This simplifies access management, as permissions are assigned to the group rather than individual users.

Roles:

IAM Roles are identities with specific permissions. Unlike users, roles are not assigned to a single person but can be assumed by trusted entities like EC2 instances or other AWS services.

Roles allow AWS resources (such as EC2 instances) to securely interact with other AWS services, and can also be assumed by users in another AWS account or a federated identity provider.

Policies:

Policies are JSON documents that define permissions. They can specify allowed or denied actions on specific resources. Policies are applied to users, groups, or roles.

Managed policies are predefined by AWS, while Inline policies are specific to a user, group, or role.

Policies include permissions such as whether a user can access an S3 bucket or launch EC2 instances.

Multi-Factor Authentication (MFA):

MFA is an added layer of security requiring users to present something they know (password) and something they have (a time-sensitive code from a physical or virtual MFA device).

Temporary Credentials:

IAM allows you to provide temporary security credentials through the use of roles, which can be assumed by applications or users, typically for limited time spans. This is often used for federated access or when applications need to interact with AWS securely.

Identity Federation:

IAM supports identity federation, allowing you to authenticate users from external sources like corporate directories (Active Directory), or social login providers (Google, Facebook). This enables Single Sign-On (SSO) to AWS resources.

Fine-Grained Access Control:

IAM provides resource-based policies and condition keys to control access based on context (e.g., IP address, time of day, or whether the request is encrypted).

Policies can be as granular as restricting access to a particular S3 bucket or allowing actions only on specific EC2 instances.

## Audit and Monitoring:

AWS CloudTrail integrates with IAM, providing logs of every API call, including who made the call and what actions were taken. This helps in monitoring and auditing permissions and detecting potential security issues.

Best Practices for Using IAM:

Enable MFA for Root and IAM Users to prevent unauthorized access.

Follow the principle of least privilege, granting only the permissions necessary for tasks.

Use roles for applications rather than embedding long-term credentials.

Regularly review permissions and remove unused access.

Use IAM groups for organizing users with similar roles, streamlining management.

Monitor IAM activity through CloudTrail for auditing and compliance purposes.

======================================================================================================================

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in AWS. It is designed to make data querying, analysis, and storage faster and more cost-effective for large datasets.

======================================================================================================================

Amazon CodePipeline is a fully managed continuous integration and continuous delivery (CI/CD) service provided by AWS. It helps automate the steps required to release software changes, enabling developers to deliver updates quickly and reliably.

======================================================================================================================

## Compute

## EC2 (Elastic Compute Cloud): Virtual servers in the cloud, scalable and customizable.

## Lambda: Serverless compute, run code without managing servers.

## ECS (Elastic Container Service): Run and manage containers using Docker.

## EKS (Elastic Kubernetes Service): Managed Kubernetes service.

## AWS Fargate: Serverless compute engine for containers.

## Lightsail: Simplified virtual servers for small projects.

## Outposts: Run AWS infrastructure on-premises.

## Batch: Run batch computing workloads efficiently.

## Storage

## S3 (Simple Storage Service): Scalable object storage for data and files.

## EBS (Elastic Block Store): Block storage for EC2 instances.

## EFS (Elastic File System): Managed scalable file storage.

## Glacier: Low-cost, long-term archival storage.

## Storage Gateway: Connects on-premises storage to AWS.

## FSx: Fully managed file systems like Windows File Server and Lustre.

## Databases

## RDS (Relational Database Service): Managed relational databases (MySQL, PostgreSQL, etc.).

## DynamoDB: Fully managed NoSQL database.

## Aurora: High-performance relational database engine.

## ElastiCache: In-memory data caching (Redis/Memcached).

## Redshift: Data warehousing service for analytics.

## DocumentDB: Managed NoSQL database for document storage (MongoDB-compatible).

## Neptune: Graph database for relationships and networks.

## Networking

## VPC (Virtual Private Cloud): Isolated virtual networks for AWS resources.

## Route 53: Scalable DNS and domain management.

## CloudFront: Content delivery network (CDN) for faster content delivery.

## Elastic Load Balancing (ELB): Distributes traffic across servers.

## Direct Connect: Dedicated network connection to AWS.

## VPN (Virtual Private Network): Securely connect on-premises networks to AWS.

## Developer Tools

## CodeCommit: Managed source control (like Git).

## CodePipeline: Automates CI/CD workflows.

## CodeBuild: Compiles code and runs tests.

## CodeDeploy: Automates deployments to EC2, Lambda, etc.

## Cloud9: Web-based development environment (IDE).

## X-Ray: Debug and analyze applications.

## Machine Learning

## SageMaker: Build, train, and deploy machine learning models.

## Rekognition: Image and video analysis.

## Comprehend: Natural language processing (NLP).

## Translate: Machine translation service.

## Polly: Text-to-speech service.

## Lex: Build conversational interfaces (chatbots).

## Personalize: Recommendations engine.

## Forecast: Predict future business outcomes.

## Analytics

## Athena: Query S3 data using SQL.

## EMR (Elastic MapReduce): Big data processing (Hadoop, Spark).

## Kinesis: Real-time data streaming and analysis.

## QuickSight: Business intelligence and visualization.

## Glue: Data integration and ETL (Extract, Transform, Load).

## Data Pipeline: Workflow orchestration for data.

## Security & Identity

## IAM (Identity and Access Management): Secure access control for AWS resources.

## Cognito: User authentication and identity federation.

## Secrets Manager: Manage and rotate secrets (API keys, passwords).

## Certificate Manager: Provision and manage SSL/TLS certificates.

## GuardDuty: Threat detection and monitoring.

## Macie: Data security and privacy monitoring.

## AWS WAF (Web Application Firewall): Protects applications from common web exploits.

## Shield: DDoS protection service.

## Application Integration

## SQS (Simple Queue Service): Message queuing service.

## SNS (Simple Notification Service): Push notifications and messaging.

## Step Functions: Workflow orchestration with visual workflows.

## EventBridge: Event-driven application integration.

## AppFlow: Data integration for SaaS apps.

## Migration & Transfer

## DMS (Database Migration Service): Migrate databases to AWS.

## Snowball/Snowmobile: Physical data transfer devices for large-scale migrations.

## Transfer Family: Securely transfer files (FTP, SFTP).

## Management & Governance

## CloudWatch: Monitoring and logging for AWS resources.

## CloudTrail: Tracks user activity and API calls.

## Config: Tracks AWS resource configurations and compliance.

## Systems Manager: Centralized management for AWS and on-premises resources.

## Service Catalog: Manage and deploy approved IT services.

## AWS Organizations: Manage multiple AWS accounts centrally.

## Trusted Advisor: Recommendations for cost optimization, security, and performance.

## IoT

## IoT Core: Connect IoT devices to the cloud.

## Greengrass: IoT edge computing.

## IoT Analytics: Analyze IoT data.

## IoT Device Defender: Secure IoT devices.

## Media Services

## MediaConvert: Convert video files for on-demand streaming.

## MediaLive: Broadcast live video streams.

## MediaPackage: Prepare and protect video streams.

## Kinesis Video Streams: Process and analyze video streams.

## AR & VR

## Sumerian: Create AR, VR, and 3D applications.

## Customer Engagement

## Connect: Cloud-based contact center solution.

## Pinpoint: Engage customers via messaging (email, SMS, push).

## Blockchain

## Managed Blockchain: Set up and manage blockchain networks.

## QLDB (Quantum Ledger Database): Immutable ledger database.

## Game Development

## GameLift: Managed service for deploying, operating, and scaling multiplayer games.

## Quantum Computing

## Braket: Explore and experiment with quantum computing.

**Comparison with Other EC2 Pricing Models**

| **Feature** | **On-Demand Instances** | **Spot Instances** | **Reserved Instances** | **Savings Plans** |
| --- | --- | --- | --- | --- |
| **Cost** | Standard rate (higher cost) | Up to 90% savings | Lower cost with commitment | Discount based on commitment |
| **Commitment** | None | None | 1 or 3 years | 1 or 3 years |
| **Interruption Risk** | None | AWS can terminate | None | None |
| **Best For** | Unpredictable, spiky workloads | Flexible, fault-tolerant workloads | Predictable, steady workloads | Predictable usage with flexibility |

Veritas Cluster

Veritas Cluster Server (VCS), now part of the Veritas InfoScale family, is a high-availability and disaster recovery solution used to manage critical applications and services. It ensures that applications remain online by automatically detecting faults and failing over to backup servers. Below are key details:

### **Core Features of Veritas Cluster**

1. High Availability:
   * Detects failures in hardware, software, or network.
   * Automatically migrates workloads to a healthy system within the cluster.
2. Scalability:
   * Supports clusters ranging from two nodes to multiple geographically distributed nodes.
3. Application Awareness:
   * Supports a wide range of applications, including databases (e.g., Oracle, SQL Server), web servers, and custom applications.
   * Predefined agents make it easier to configure for specific applications.
4. Global Clustering:
   * Enables disaster recovery by spanning clusters across different geographical locations.
5. Dynamic Storage Management:
   * Works seamlessly with Veritas Volume Manager to manage and mirror data across storage systems.
6. Intuitive Interface:
   * Comes with graphical user interfaces (GUIs) and command-line tools for configuration and management.

### **Comparison Table**

| **Feature** | **Active-Active** | **Active-Passive** |
| --- | --- | --- |
| Node Usage | All nodes active | Only one node active at a time |
| Performance | Balanced across nodes | Depends on a single active node |
| Failover | Load redistributed among nodes | Passive node takes over |
| Complexity | Higher (requires load balancing) | Lower (simple to configure) |
| Best For | Stateless, distributed services | Stateful, single-instance services |

I/O Fencing (Input/Output Fencing) is a critical feature in high-availability clusters, such as those managed by Veritas Cluster Server (VCS). It ensures data integrity during scenarios like split-brain, where nodes lose communication but continue to operate independently. In such cases, I/O fencing prevents multiple nodes from simultaneously accessing the shared storage, which could lead to data corruption.

Core Components of Veritas Cluster Server

1. Cluster Communication Modules:
   * Low-Latency Transport (LLT): Facilitates efficient and reliable communication between cluster nodes.
   * Group Membership and Atomic Broadcast (GAB): Ensures consistent cluster membership and reliable message delivery between nodes.
2. Cluster Engine:
   * High Availability Daemon (HAD): The main daemon that manages resources and service groups within the cluster.
3. Service Groups:
   * Logical containers grouping related resources (e.g., storage, IP, application). Service groups can fail over as a unit to another node.
4. Resources:
   * Individual entities (e.g., a disk, network interface, application) managed by the cluster. Each resource is configured with an agent to handle specific operations like start, stop, and monitor.
5. Agents:
   * Software modules that enable the cluster to manage and monitor resources. Agents are classified as:
     + Bundled Agents: Pre-installed agents for common resources like storage, IPs, and file systems.
     + Custom Agents: User-defined agents for specific applications or services.
6. Cluster Configuration Files:
   * Main.cf: Defines the configuration of the cluster, including service groups and resources.
   * Types.cf: Contains the resource type definitions and agent specifications.
7. I/O Fencing:
   * Protects shared storage from being accessed by multiple nodes in a split-brain scenario. It uses coordinator disks or Coordination Point Servers (CP servers) for arbitration.
8. Cluster Membership and Quorum:
   * Ensures that only nodes with quorum (majority or specific authority) can run cluster services.
9. Coordination Points:
   * Devices or servers (e.g., coordinator disks or CP servers) used for quorum enforcement in fencing.

### **Key Differences**

| **Aspect** | **Failover Service Group** | **Parallel Service Group** |
| --- | --- | --- |
| Nodes Active | Only one node at a time | Multiple nodes simultaneously |
| Failover | Fails over to another node if necessary | No failover; workload continues on other nodes |
| Resource Sharing | Resources are exclusive to one node | Resources are shared across nodes |
| Application Type | Non-parallel applications (e.g., databases) | Parallel applications (e.g., web servers) |
| Scalability | Limited to one active node | Scalable across multiple nodes |

How to Add IP V4 NMCLI

nmcli con add con-name enp0s9 type ethernet ifname enp0s9 ipv4.method manual ipv4.address 192.168.1.11/24

Veritas installer configuration cluster

22 nmcli add con-name enp0s8 type ethernet ifname enp0s8 ipv4.method manual ipv4.address 192.168.1.10/24

23 nmcli con add con-name enp0s8 type ethernet ifname enp0s8 ipv4.method manual ipv4.address 192.168.1.10/24

24 ifconfig

25 nmcli con add con-name enp0s8 type ethernet ifname enp0s8 ipv4.method manual ipv4.address 192.168.1.11/24

26 nmcli con add con-name enp0s9 type ethernet ifname enp0s9 ipv4.method manual ipv4.address 192.168.1.11/24

27 ifconfig

28 nmcli con show

29 history

30 ssh root@192.168.1.14

31 ifconfig

32 ping 192.168.1.14

33 ping 192.168.1.13

34 clear

35 hostname

36 ifconfig

37 vi /etc/hosts

38 cat /etc/hosts

39 ping server2.localhost.com

40 ifconfig

41 hostname

42 vi /etc/hosts

43 ping server2.localhost.com

44 ll

45 Veritas\_InfoScale\_7.4.2\_CentOS

46 ll

47 cd Veritas\_InfoScale\_7.4.2\_CentOS

48 ll

49 cd dvd1-redhatlinux

50 ll

51 cd rhel7\_x86\_64

52 ll

53 ./installer

54 sudo ./installer

55 pwd

56 ll

57 installer

58 cd ..

59 ll

60 cd ..

61 ll

62 cd ..

63 ll

64 chmod 777 Veritas\_InfoScale\_7.4.2\_CentOS/

65 cd Veritas\_InfoScale\_7.4.2\_CentOS/dvd1-redhatlinux/

66 ll

67 cd rhel7\_x86\_64

68 ll

69 ./installer

70 cd ..

71 ll

72 chmod 777 Veritas\_InfoScale\_7.4.2\_CentOS/ -R

73 cd -

74 ll

75 cd dvd1-redhatlinux/

76 ll

77 cd rhel7\_x86\_64

78 ll

79 ./installer

80 clear

81 ./installer

82 ./installer -configure

83 pwd

84 ll

85 cd /opt/

86 ll

87 cd VRTSvcs

88 ll

89 cd bin/

90 ll

91 hastatus

92 ./hastatus

93 ./hastatus -sum

94 ll

95 ls

96 cat /etc/llttab

97 cd /etc/VRTSvcs

98 ll

99 cd conf/config/

100 ll

101 pwd

102 vi main.cf

103 pwd

104 cd /opt/VRTSvcs/bin/

105 ll

106 ./hastatus

107 pwd

108 vi ~/.bash\_profile

109 hastatus

110 source ~/.bash\_profile

111 hastatus

112 cat ~/.bash\_profile

Cluster information verification:

Cluster Name: cluster1

Cluster ID Number: 12114

Private Heartbeat NICs for server1.localhost.com:

link1=enp0s8

link2=enp0s9

Low-Priority Heartbeat NIC for server1.localhost.com:

link-lowpri1=enp0s3

Private Heartbeat NICs for server2.localhost.com:

link1=enp0s8

link2=enp0s9

Low-Priority Heartbeat NIC for server2.localhost.com:

link-lowpri1=enp0s3

Cluster configuration information:

Cluster Name: cluster1

Cluster ID Number: 12114

Private Heartbeat NICs for server1.localhost.com:

link1=enp0s8

link2=enp0s9

Low-Priority Heartbeat NIC for server1.localhost.com:

link-lowpri1=enp0s3 Private Heartbeat NICs for server2.localhost.com:

link1=enp0s8

link2=enp0s9

Low-Priority Heartbeat NIC for server2.localhost.com:

link-lowpri1=enp0s3

Veritas InfoScale Enterprise Shutdown completed successfully

The following processes were stopped on server1.localhost.com:

vcsmm

vxgms

vxglm

vxcpserv

had

[root@server1 bin]# vi ~/.bash\_profile

PATH=$PATH:$HOME/bin:/opt/VRTSvcs/bin:/opt/VRTS/bin

source ~/.bash\_profile

[root@server1 bin]# haconf -makerw

[root@server1 bin]# hagrp -add HAShare

VCS NOTICE V-16-1-10136 Group added; populating SystemList and setting the Parallel attribute recommended before adding resources

[root@server1 bin]# hagrp -modify HAShare SystemList server1.localhost.com 0 server2.localhost.com 1

VCS WARNING V-16-1-10176 System server1.localhost.com not defined: cannot modify SystemList

[root@server1 bin]# hasys -list

server1

server2

[root@server1 bin]# hagrp -modify HAShare SystemList server1 0 server2 1 [root@server1 bin]# hasys -list

server1

server2

[root@server1 bin]# hagrp -modify HAShare AutoStartList server1 server2

[root@server1 bin]# hagrp -list

ClusterService server1

ClusterService server2

HAShare server1

HAShare server2

[root@server1 bin]# hastatus -sum

-- SYSTEM STATE

-- System State Frozen

A server1 RUNNING 0

A server2 RUNNING 0

-- GROUP STATE

-- Group System Probed AutoDisabled State

B ClusterService server1 Y N ONLINE

B ClusterService server2 Y N OFFLINE

B HAShare server1 Y N OFFLINE

B HAShare server2 Y N OFFLINE

[root@server1 bin]# vxdisk list

DEVICE TYPE DISK GROUP STATUS

server1\_disk\_0 auto:none - - online invalid

server1\_disk\_1 auto:LVM - - LVM

[root@server1 bin]# vxdisk list

DEVICE TYPE DISK GROUP STATUS

server1\_disk\_0 auto:none - - online invalid

server1\_disk\_1 auto:LVM - - LVM

[root@server1 bin]# vxdisksetup -i server1\_disk\_0

[root@server1 bin]# vxdisk list

DEVICE TYPE DISK GROUP STATUS

server1\_disk\_0 auto:cdsdisk - - online

server1\_disk\_1 auto:LVM - - LVM

Server 2 :   
  
[root@server2 ~]# vxdctl enable

[root@server2 ~]# vxdisk list

DEVICE TYPE DISK GROUP STATUS

server2\_disk\_0 auto:LVM - - LVM

server2\_disk\_1 auto:cdsdisk - - online

[root@server1 bin]# hares -add HAShare-ip IP HAShare

VCS NOTICE V-16-1-10242 Resource added. Enabled attribute must be set before agent monitors

[root@server1 bin]# hares -modify HAShare-ip Enabled 1

[root@server1 bin]# hares -add HAShare-ip IP HAShare

VCS NOTICE V-16-1-10242 Resource added. Enabled attribute must be set before agent monitors

[root@server1 bin]# hares -modify HAShare-ip Enabled 1

[root@server1 bin]# hares -modify HAShare-ip Address "192.168.1.18"

[root@server1 bin]# hares -modify HAShare-ip NetMask "255.255.255.0"

root@server1 bin]# hares -modify HAShare-ip Device "enp0s3"

[root@server1 bin]# hares -list

HAShare-ip server1

HAShare-ip server2

csgnic server1

csgnic server2

webip server1

webip server2

[root@server1 bin]# haconf -makerw

[root@server1 bin]# hares -add DG1 DiskGroup HAShare

VCS NOTICE V-16-1-10242 Resource added. Enabled attribute must be set before agent monitors

[root@server1 bin]# hares -modify DG1 DiskGroup DG1

[root@server1 bin]#

[root@server1 bin]# hares -modify HAShare-mount MountPoint /hashare\_mnt/

[root@server1 bin]# hares -modify HAShare-mount BlockDevice /dev/vx/dsk/DG1/hashare\_lv

[root@server1 bin]# hares -modify HAShare-mount FsckOpt "%-y"

[root@server1 bin]#

[root@server1 bin]# hares -list

DG1 server1

DG1 server2

HAShare-ip server1

HAShare-ip server2

HAShare-mount server1

HAShare-mount server2

HAShare-nic server1

HAShare-nic server2

csgnic server1

csgnic server2

webip server1

webip server2

[root@server1 bin]# haconf -makerw

[root@server1 bin]# hagrp -enableresources HAShare

[root@server1 bin]# hares -online HAShare-mount -sys server1

[root@server1 ~]# hastatus -sum

-- SYSTEM STATE

-- System State Frozen

A server1 RUNNING 0

A server2 RUNNING 0

-- GROUP STATE

-- Group System Probed AutoDisabled State

B ClusterService server1 Y N OFFLINE

B ClusterService server2 Y N ONLINE

B HAShare server1 Y N ONLINE

B HAShare server2 Y N OFFLINE

[root@server1 hashare\_mnt]# hagrp -switch HAShare -to server2

[root@server1 hashare\_mnt]# hastatus -sum

-- SYSTEM STATE

-- System State Frozen

A server1 RUNNING 0

A server2 RUNNING 0

-- GROUP STATE

-- Group System Probed AutoDisabled State

B ClusterService server1 Y N OFFLINE

B ClusterService server2 Y N ONLINE

B HAShare server1 Y N OFFLINE

B HAShare server2 Y N ONLINE

The sudo command in Linux and other Unix-like operating systems stands for "superuser do." It allows a regular user to execute commands with the privileges of another user (usually the superuser, also called root), which has administrative rights.

Regenerate missing kernel image

dracut initramfs-3.10.0-1160.el7.x86\_64.img 3.10.0-1160 l7.x86\_64

mkinitrd initramfs-3.10.0-1160.el7.x86\_64.img 3.10.0-1160.el7.x86\_64 --force

GRUB rescue

grub-install /dev/sda

grub-mkconfig -o /boot/grub/grub.cfg

Press ‘**e**’ to enter in the edit mode and then go to the end of line which starts with ‘**linux**‘ word, type the keyword “**rd.break**”

switch\_root:/# mount -o remount,rw /sysroot

switch\_root:/# chroot /sysroot

# **CentOS 7 to CentOS 8 Stream Migration**

sudo yum update -y

yum install epel-release -y

yum install rpmconf -y

rpmconf -a

Simplify package management and repository handling for transitioning from CentOS 7 to CentOS 8 by install yum-utils which is needed for removing packages:

sudo yum install yum-utils -y

sudo package-cleanup --leaves

sudo package-cleanup --orphans

After successful removal reboot the system!

CentOS 8 and distributions based on RHEL 8 have moved from using [yum](https://en.wikipedia.org/wiki/Yum_(software)) package manager as their primary package management tool to [dnf](https://en.wikipedia.org/wiki/DNF_(software)) package manager. Therefore yum is no longer needed for us and we now need dnf. When the server becomes online again after the reboot, access the server and remove yum followed by installing dnf.

sudo yum install dnf -y

sudo dnf remove yum yum-metadata-parser -y

sudo rm -Rf /etc/yum

Now, upgrade your CentOS 7 server to CentOS 8 with dnf package manager:

sudo dnf update -y

After updating, install CentOS 8 release package and upgrade with the installed package:

sudo dnf install http://vault.centos.org/8.5.2111/BaseOS/x86\_64/os/Packages/{centos-linux-repos-8-3.el8.noarch.rpm,centos-linux-release-8.5-1.2111.el8.noarch.rpm,centos-gpg-keys-8-3.el8.noarch.rpm} -y

sudo dnf upgrade https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y

KVM installation

[root@localhost ~]# dnf install qemu-kvm qemu-img libvirt virt-install -y

Last metadata expiration check: 0:02:37 ago on Friday 29 November 2024 04:45:42 PM.

Dependencies resolved.  
  
[root@localhost ~]# virt-host-validate

QEMU: Checking for hardware virtualization : FAIL (Host not compatible with KVM; HW virtualization CPU features not found. Only emulated CPUs are available; performance will be significantly limited)

QEMU: Checking if device '/dev/vhost-net' exists : PASS

QEMU: Checking if device '/dev/net/tun' exists : PASS

QEMU: Checking for cgroup 'cpu' controller support : PASS

QEMU: Checking for cgroup 'cpuacct' controller support : PASS

QEMU: Checking for cgroup 'cpuset' controller support : PASS

QEMU: Checking for cgroup 'memory' controller support : PASS

QEMU: Checking for cgroup 'devices' controller support : PASS

QEMU: Checking for cgroup 'blkio' controller support : PASS

QEMU: Checking for device assignment IOMMU support : WARN (Unknown if this platform has IOMMU support)

QEMU: Checking for secure guest support : WARN (Unknown if this platform has Secure Guest support)

dnf install cockpit cockpit-machines

[root@kvm-01 ~]# systemctl enable --now cockpit.socket

Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/systemd/system/cockpit.socket.

[root@kvm-01 ~]#

sudo zypper install <package\_name>

sudo zypper install vim

sudo zypper install <package\_name1> <package\_name2>

zypper remove <package\_name>

zypper remove --clean-deps <package\_name>

zypper list-updates

zypper search apache2

Mutipath

Rescan SCSI Bus for New Devices

echo "- - -" > /sys/class/scsi\_host/hostX/scan

For the configuration details of multipath:

bash

Copy code

cat /etc/multipath.conf

Flush Multipath Devices

Remove a Multipath Device

multipath -d <device\_name> remove device

multipath -f <device\_name> flush device

multipath -r reload multipath

restart services

systemctl start multipathd

Summary of Runlevels:

| **Runlevel** | **Name** | **Description** |
| --- | --- | --- |

| 0 | Halt | Power off the system |
| --- | --- | --- |

| 1 | Single-user mode | Maintenance mode for system repair or recovery |
| --- | --- | --- |

| 2 | Multi-user without network | Multi-user mode, no network services |
| --- | --- | --- |

| 3 | Multi-user with network | Multi-user mode with network services |
| --- | --- | --- |

| 4 | Unused/Custom | Can be customized by the user or sysadmin |
| --- | --- | --- |

| 5 | Multi-user with GUI | Multi-user mode with a graphical interface |
| --- | --- | --- |

| 6 | Reboot | Reboot the system |
| --- | --- | --- |

User Management commands

useradd john

useradd -m -d /home/john -s /bin/bash john

passwd john

userdel -r john

usermod -L john locking

usermod -U john unlocking

groupadd developers add group

usermod -aG developers john add secondary group

sudo gpasswd -d john developers remove user from group

groupdel developers

usermod -u 1500 john change UID change

groupmod -g 1500 developers GID change

usermod -e 2025-12-31 john set expiry

useradd -m -d /home/john -s /bin/bash -G developers john all in one

Install Ansible

**sudo yum install ansible**

**dnf install ansible**

### **Prepare Your Inventory**

**Create an inventory file (e.g., inventory.ini) that lists your target hosts. This file can be structured like:**

**ini**

**Copy code**

**[linux\_servers]**

**server1 ansible\_host=192.168.1.10**

**server2 ansible\_host=192.168.1.11**

ansible linux\_servers -i inventory.ini -m yum -a "name=\* state=latest" --become

Create a file called patching.yml:

---

- name: Patching Linux Servers

hosts: linux\_servers

become: yes # Run as root/sudo

tasks:

- name: Update APT cache (Debian-based)

apt:

update\_cache: yes

cache\_valid\_time: 3600

when: ansible\_facts['pkg\_mgr'] == 'apt'

- name: Upgrade all packages (Debian-based)

apt:

upgrade: dist

autoclean: yes

autoremove: yes

when: ansible\_facts['pkg\_mgr'] == 'apt'

- name: Update YUM packages (RedHat-based)

yum:

name: "\*"

state: latest

when: ansible\_facts['pkg\_mgr'] == 'yum'

- name: Clean YUM cache (RedHat-based)

yum:

name: yum

state: latest

update\_cache: yes

when: ansible\_facts['pkg\_mgr'] == 'yum'

Run the playbook:

bash

Copy code

ansible-playbook -i inventory.ini patching.yml

Crontab  
[Minute] [Hour] [Day of the Month] [Month] [Day of the Week] [Command]

Upgrading from RHEL 8 to RHEL 9

#### Step 1: Remove “tmp\_leapp\_py3” Directory

#### Lock the System to RHEL 8.6

subscription-manager release --set 8.6

dnf update

#### Install the Leapp Utility

dnf install leapp-upgrade -y

#### Remove Versionlock Plugin

dnf versionlock clear

disable **AllowZoneDrifting**

nano /etc/firewalld/firewalld.conf

#### Perform the Pre-upgrade Phase

leapp preupgrade --target 9.0

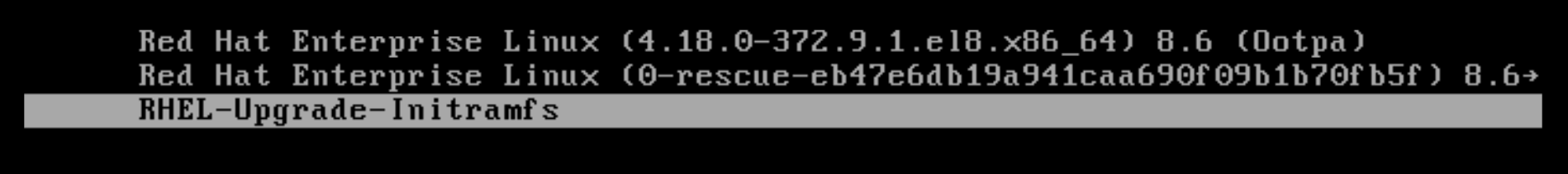
#### Upgrading from RHEL 8 to RHEL 9

leapp upgrade --target 9.0

reboot

#### Step 12: Choose RHEL-Upgrade-initramfs

Once you reboot, choose the third option labeled “**RHEL-Upgrade-initramfs**”.



Offline run from ISO and local.repo

sudo rpm -Uvh /path/to/dnf-plugin-system-upgrade.rpm sudo rpm -Uvh /path/to/dnf.rpm sudo rpm -Uvh /path/to/python3-dnf-plugin.rpm

dnf system-upgrade download --releasever=9 --setopt=deltarpm=false

dnf system-upgrade reboot

Server Hardening

Update and Patch the Server Regularly

Minimize Installed Software

Configure Firewalls

Secure Remote Access

Enable SELinux

Expiry user account set

Use Disk encryption

Disable unnecessary ports

### **Relationship Between VA and SCD**

* **Vulnerability Assessment (VA)** is the process of identifying security risks in the system. Once vulnerabilities are identified, the next step is addressing them.
* **System Configuration Detection (SCD)** and **Secure Configuration Deployment** come into play to ensure that systems are configured in a way that minimizes vulnerabilities. They help in ensuring that the systems adhere to security standards and best practices, and that the configurations are locked down to prevent potential security holes.

Linux Network Bonding: An Overview

Network Bonding in Linux is a technique where multiple network interfaces (NICs) are combined into a single logical interface to improve network performance, redundancy, or both. This can be used for various purposes, such as increasing bandwidth, providing failover support in case of a network interface failure, or improving network reliability.

Linux provides a bonding driver that allows you to combine two or more network interfaces (such as Ethernet) into a single bonded interface. This bonded interface is then treated as a single network interface by the operating system.

**Round robin**: traffic goes through all active ports: **Failover**: If one slave fails, traffic is sent through the remaining active interfaces.

**Mode 1 – Active/Backup**: **Failover**: If the active interface fails, another interface becomes active.

# **Configure NIC(Network Interface Card) bonding in CentOS 7 / RHEL 7**

If bonding module is not loaded on your linux box then use the below command to load.

[root@openstack ~]# modprobe bonding

To list the bonding module info, use following command.

[root@openstack ~]# modinfo bonding

##### Step:1 Create Bond Interface File

Create a bond interface file (**ifcfg-bond0**) under the folder “**/etc/sysconfig/network-scripts/**”

[root@openstack network-scripts]# vi ifcfg-bond0

DEVICE=bond0

TYPE=Bond

NAME=bond0

BONDING\_MASTER=yes

BOOTPROTO=none

ONBOOT=yes

IPADDR=192.168.1.70

NETMASK=255.255.255.0

GATEWAY=192.168.1.1

BONDING\_OPTS="mode=5 miimon=100"

##### Step:2 Edit the NIC interface files

**For ifcfg-enp0s3**

[root@openstack ~]# vi /etc/sysconfig/network-scripts/ifcfg-enp0s3

TYPE=Ethernet

BOOTPROTO=none

DEVICE=enp0s3

ONBOOT=yes

HWADDR="08:00:27:69:60:c9"

MASTER=bond0

SLAVE=yes

**For ifcfg-enp0s8**

[root@openstack ~]# cat /etc/sysconfig/network-scripts/ifcfg-enp0s8

TYPE=Ethernet

BOOTPROTO=none

DEVICE=enp0s8

ONBOOT=yes

HWADDR="08:00:27:ea:71:8d"

MASTER=bond0

SLAVE=yes

##### Restart the Network Service

systemctl restart network.service

[root@openstack ~]# cat /proc/net/bonding/bond0

DEVOPS LIFE CYCLE  
  
**Plan**

* Goal: Define the software development process and objectives.
* Activities:
  + Requirement gathering.
  + Project roadmap creation.
  + Use tools like JIRA, Trello, or Azure Boards for task management.
* Key Output: A clear blueprint for the project.

2. Develop

* Goal: Write and test the application code.
* Activities:
  + Coding using version control systems like Git or Bitbucket.
  + Peer reviews to ensure quality.
  + Use tools like GitHub and GitLab for source code management.
* Key Output: Well-documented, versioned code.

3. Build

* Goal: Compile code and resolve dependencies to create a deployable artifact.
* Activities:
  + Use build tools like Maven, Gradle, or Ant to compile and package the code.
  + Automate builds with Jenkins, Travis CI, or CircleCI.
* Key Output: A validated and deployable build artifact.

4. Test

* Goal: Ensure the software is bug-free and meets requirements.
* Activities:
  + Conduct automated and manual tests (unit, integration, performance, etc.).
  + Use tools like Selenium, JUnit, TestNG, or SonarQube for quality checks.
* Key Output: A reliable software build.

5. Release

* Goal: Prepare the software for deployment to production.
* Activities:
  + Automate release pipelines using tools like Jenkins, GitLab CI/CD, or Azure DevOps.
  + Versioning and tagging of releases.
* Key Output: A packaged release artifact ready for deployment.

6. Deploy

* Goal: Deliver the application to the production environment.
* Activities:
  + Use configuration management tools like Ansible, Chef, or Puppet for consistent deployments.
  + Implement containerization with Docker and orchestration using Kubernetes.
* Key Output: Deployed software in a live environment.

7. Operate

* Goal: Monitor and manage the application in production.
* Activities:
  + Use monitoring tools like Nagios, Prometheus, Grafana, or Splunk for system health tracking.
  + Automate scaling and updates.
* Key Output: Reliable and scalable application operations.

8. Monitor

* Goal: Continuously track application and infrastructure performance.
* Activities:
  + Log analysis using tools like ELK Stack (Elasticsearch, Logstash, Kibana).
  + Performance monitoring to identify bottlenecks.
  + Collect user feedback for future improvements.
* Key Output: Insights for optimization and improvement.

Ansible Tower setup guide

1.update dnf or yum and set hostname ansible.tower

2. Register for ansible tower subscription

3. Generate ssh-keygen -t rsa

[root@ansible ~]# ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa): /root/.ssh/id\_rsa

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa

Your public key has been saved in /root/.ssh/id\_rsa.pub

The key fingerprint is:

SHA256:pPPt1i7+IZ1iI1WXGNBkr9ffHSJreDgx+wXU1fwAsu8 root@ansible.tower

The key's randomart image is:

+---[RSA 3072]----+

| o+=. o.|

| +o+o.o|

| . ..o.+..|

| o .o o ..|

| o So.oo....|

| o o\*o+o. =|

| o=\*=E. +|

| +B+o. |

| oo=o |

+----[SHA256]-----+

[root@ansible ~]# ssh-copy-id -i ~/.ssh/id\_rsa.pub root@192.168.222.131

/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id\_rsa.pub"

The authenticity of host '192.168.222.131 (192.168.222.131)' can't be established.

ED25519 key fingerprint is SHA256:eEAZwtMzDgSSJaHLB5czAMSSQyIEUDGBP2eV7bU6JzU.

This key is not known by any other names

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

[root@ansible ansible]# cat /etc/ansible/hosts

# This is the default ansible 'hosts' file.

#

# It should live in /etc/ansible/hosts

#

# - Comments begin with the '#' character

# - Blank lines are ignored

# - Groups of hosts are delimited by [header] elements

# - You can enter hostnames or ip addresses

# - A hostname/ip can be a member of multiple groups

[web]

server1 ansible\_ssh\_host=192.168.222.131 ansible\_ssh\_port=22 ansible\_ssh\_user=root

server2 ansible\_ssh\_host=192.168.222.132 ansible\_ssh\_port=22 ansible\_ssh\_user=root

Command from main line

[root@ansible ~]# ansible -m shell -a "free -m" web

server2 | CHANGED | rc=0 >>

total used free shared buff/cache available

Mem: 1021 369 594 4 203 652

Swap: 0 0 0

4. Create playbook under project folder > playbook.yaml

- name: Install Packages

hosts:

- web

- database

tasks:

- name: Install php and nginx

package:

name:

- php

- httpd

state: present

Run : ansible-playbook playbook.yaml

[root@ansible project]# cat playbook.yaml

- name: Install Packages

hosts:

- web

tasks:

- name: Install php and nginx

package:

name:

- php

- httpd

state: present

[root@ansible project]# cat dnf\_update.yaml

- name: Update all packages on managed nodes

hosts: all

become: true # Ensures the playbook runs with elevated privileges

tasks:

- name: Update all packages to the latest version

dnf:

name: "\*"

state: latest

- name: Clean up unused packages (optional)

dnf:

autoremove: yes

[root@ansible project]# cat reboot.yaml

- name: Reboot all inventory hosts

hosts: all

become: true # Run tasks as root

tasks:

- name: Reboot the system

reboot:

reboot\_timeout: 300 # Adjust timeout as needed

GIT setup

mkdir GIT (main directory)

* git init to make the GIT main branch directory
* git add filename.txt to add tracking
* git commit to add snapshot to the master branch
* git status to see if any changes are committed
* git log for last 10 commits

##### $ git log

##### commit b5ab9544c9a34b4428f40f28e4b14ae39c5bb344 (HEAD -> master)

##### Author: tusharnvgf <tusharthegamer@gmail.com>

##### Date: Sun Dec 8 07:49:06 2024 +0530

##### 

##### tusha@Tushar-StudyPC MINGW64 /c/GIT/Gitraining\_v1 (master)

##### $ git log --author "tusharnvgf"

##### commit b5ab9544c9a34b4428f40f28e4b14ae39c5bb344 (HEAD -> master)

##### Author: tusharnvgf <tusharthegamer@gmail.com>

##### Date: Sun Dec 8 07:49:06 2024 +0530

##### 

##### 