**Devops Notes from Siva Notes**

**Date: 11/02/2025**

**# ssh-keygen -f <file-name>**

**# ssh -i <path-to-private-key> username@IP**

**# ssh -i private-key/password username@IP**

**# rm -r devops --> recursive --> go inside every folder and delete everything**

**# grep <word-to-find> <file-name>**

**# | --> piping one command output will become input to another command**

**# wget --> download files**

**# curl --> downloads the text content directly on to terminal**

**# awk command is used to divide the data based on columns**

**# head <file-name> --> first 10lines**

**# tail <file-name< --> last 10 lines**

**VIM**

**# :/<word-to-search> --> search from top**

**# :?<word-to-search> --> search from bottom**

**# :s/<word-to-find>/<word-to-replace> --> replace the word where your cursor is, this will replace only first occurence in that line**

**# :2s/<word-to-find>/<word-to-replace>**

**# :%s/<word-to-find>/<word-to-replace>/g --> all occurrences**

**Permissions**

**- rw- rw- r--**

**user group others**

**u g o**

**R -> 4 , W -> 2 , X -> 1 -> to run commands and shell scripts**

**sivakumar --> DevOps Team**

**user=owner**

**group=devops**

**others=testing,development team**

**in Linux when you create user, a group with same name will be created**

**sivakumar --> user sivakumar and group with name sivakumar also created**

**chmod u+x sample**

**chmod ugo+r sample**

**u --> RWX**

**g --> R**

**o -> 0**

**740**

**u --> RWX**

**g --> RX**

**o --> X**

**751**

**User management**

**1. create user**

**# useradd <user-name>**

**#passwd <user-name>**

**#/etc/passwd --> will have user entries**

**when a user is created, automatically group is created with same username**

**#getent groups or less /etc/group or cat /etc/group**

**/etc/ssh/sshd\_config --> enable password authentication**

**sshd -t --> checks for syntax of the file**

**systemctl restart sshd -->**

**2. group create**

**# groupadd <group-name>**

**3. every user will have a primary group and secondary group**

**ramesh should be added to devops group**

**# usermod -g devops ramesh**

**chown --> even file owner can't run this commands**

**only sudo user can change the ownership**

**chown <user>:<group> file-name**

**chown <user>:<group> -R folder**

**4. we will create testing group, we will give ramesh secondary access to testing**

**usermod -aG testing ramesh**

**gpasswd -d ramesh testing --> deletes ramesh from testing group**

**user is leaving organisation**

**------------------------------**

**userdel <user-name> --> removes user from linux**

**first user should be removed from the group**

**1. project release**

**2. company release**

**foreground process --> block terminal**

**background process --> runs in background &**

**when process stucks we need to kill it**

**kill PID --> process may or may not be terminated**

**when process stuck even kill can't kill it**

**kill -9 --> forcefully terminate**

**package management**

**yum command is used to install packages --> RHEL = centos = fedora = AWS Linux 2**

**amazon-linux-extras**

**yum install package-name**

**apt-get**

**raheem joined, how can you enable ssh accesss through private key**

**------------------------------------------------------**

**1. create raheem user**

**2. give me your public key, keep your private key securely.**

**700**

**user --> RWX**

**group**

**Others**

**chown user:group <file-name>**

**chmod o+r <file-name>**

**Service Management**

**Algorithm**

**-----------------**

**1. key gen**

**2. public key import**

**3. firewall create**

**4. instance create**

**5. ssh connect --> private-key**

**request reach IP address**

**ssh --> 22**

**system will check anything is running ssh process with port no 22**

**now authentication will be checked**

**service --> offering service**

**systemctl start nginx --> nginx will start run**

**systemctl status nginx**

**systemctl stop ngix**

**systemctl enable nginx --> if services are enabled, after restart automatically services will run**

**systemctl disable nginx**

[**http://52.90.162.131:80**](http://52.90.162.131:80)

**Network management**

**netstat --> network statistics**

**netstat -lntp**

**troubleshooting**

**---------------------**

**system resources**

**cpu memory -->**

**HD full --> df -hT**

**RAm full --> free**

**process is running or not --> ps -ef | grep nginx**

**port opened or not --> netstat -lntp**

**systemctl status <service> -->**

**firewall is opened or not**

**How to give access to Linux user**

**---------------------------------**

**Linux Admin --> full access to linux**

**DevOps Admin --> limited sudo access**

**yum, systemctl**

**ramesh --> admin, add him to wheel group or give admin group full access**

**suresh --> devops**

**/etc/sudoers --> edit**

**visudo -->**

**%admin ALL=(ALL) /usr/bin/yum,/usr/bin/systemctl**

**/etc/sudoers.d/**

**Ngnix**

**------------**

**/usr/share/nginx/html**

**front-end servers = http servers = 80**

**hosts html/js based applications**

**back-end servers = http = 8080**

**tomcat, jboss, etc**

**java, .NET, Python**

**login.html -->** [**http://joindevops.online:8080/api/login**](http://joindevops.online:8080/api/login)

**Linux Filesystem structure**

**/ --> root folder for liux server**

**boot --> when server is getting started, server refers this directory and config from grub.cfg**

**dev --> devices --> external devices are mounted here**

**etc --> extra configuration**

**home --> users home directory are created here**

**lib/lib64 --> system needs libraries**

**/media --> cd/dvd drive**

**/mnt --> extra disks mount here**

**/opt --> optional , manual installation of services or applications will be downloaded here**

**/proc --> running process will store here with their PID**

**/proc/cpuinfo**

**/proc/meminfo**

**/root --> root user home folder**

**/run --> when server starts, server uses this directory as storage**

**/bin --> cat,vim, touch, cd**

**/sbin --> system commands like root access commands**

**/tmp --> temp files, not important**

**/usr --> all users,**

**/var --> variables --> log files,**

**inbound --> incoming traffic**

**outbound --> outbound --> traffic originates from our server**

**NAT --> Network Address Translation**

**Roboshop configure**

**1. Web/frontend tier**

**2. App/Api/Backend tier**

**3. DB tier**

**DB apps are called statefull**

**App/Web apps are called stateless**

**nginx is used as reverse proxy here.**

**devops-practice based on centos-8**

**Password: centos/DevOps321**

**Web**

**-----------**

**http = web server**

**nginx = 80**

**forward proxy vs reverse proxy**

**forward proxy : -**

**VPN --> virtual private network**

**Servers are not aware of clients are behind VPN**

**anonymous access --> hiding our identity**

**forward proxies can't understand real clients**

**security --> companies can impose restrictions not to use particular sites, file upload and to monitor use internet behaviour**

**clients are aware of proxy, servers are not awareot aware**

**reverse proxy**

**-------------------------------**

**clients are not aware of server side**

**apps use reverse proxy to secure their code**

**load balancing**

**HTTP status**

**2\*\* --> success**

**3\*\* --> redirect/temp --> images, gifs are redirected**

**failure responses:-**

**-----------------**

**400, 404, 403, 402**

**4\*\* --> client side error**

**5\*\* --> server side error --> purely project error**

**Domain registars**

**-----------------**

**godaddy**

**hostinger**

**AWS**

**hostinger --> daws76s.online**

**hostinger responsibility is to update, sivakumar reddy with details bought daws76s.online**

**and the nameservers are**

**ns1.dns-parking.com**

**ns2.dns-parking.com**

**Register a domain**

**create hostedzone in AWS**

**change nameservers**

**reconfigure roboshop web, catalogue, mongodb using domain**

**errors**

**web --> roboshop.conf catalogue IP**

**catalogue --> is running or not**

**data loaded to mongodb or not**

**catalogue.service mongodb IP is good or not**

**mongodb --> remote 0.0.0.0 allowed or not**

**firewalls**

**catalogue --> mongodb**

**How to install application in Linux**

**We should have server**

**nodejs, .net, java, python, php, go, etc.**

**install that programming langugage**

**creating a directory**

**creating a seperate user for that application**

**downloading the application**

**install dependencies**

**nodejs --> npm**

**java --> mvn**

**python --> pip**

**.net --> nuget**

**we are giving how to run in /etc/systemd/system/application.service**

**systemctl start application**

**systemctl enable application**

**nodejs --> npm**

**maven --> automatically maven will install java**

**npm install = mvn clean package = it will give a jar file**

**java --> compile --> bytecode(.jar file)**

**python=nodejs = no need of compile = no bytecode**

**t3.medium --> mongodb, shipping, mysql**

**GIT**

**10 devops engineers**

**--------------------**

**one central location to do and share the things**

**Git --> global information tracking system**

**version control --> 10 days**

**after 10 days --> I wanted to see what was script at 2nd day..**

**we should always go back to previous version if something goes wrong**

**track the changes --> through git we can understand who did that and why**

**review the changes --> our team members should review**

**backup and security**

**branching --> create a branch from master, do the changes, if everything is good then proceed**

**collabaration**

**GIT is a popular decentralised version control system**

**------------------**

**decentralised/distributed vs centralised**

**git repo --> store the code**

**repository --> some storage just like folder with some extra capabilities**

**https://github.com/daws-76s/test-repo.git**

**push --> authentication**

**username/password --> frequently we need to enter password every time**

**username/key --> it will be automatically taken from private key**

**place your public key in the server**

**keep private key with you**

**.ssh --> usually in user directory**

**/c/Users/sivakumar/.ssh**

**config --> no extensions to this file**

**git --> concept**

**github --> they took that concept and created github**

**gitlab**

**bitbucket**

**https://github.com/daws-76s/test-repo.git --> username and password**

**git@github.com:daws-76s/test-repo.git --> just private key**

**~ --> your home folder**

**/c/Users/ramesh**

**/c/Users/user/daws-76s-github**

**Host github.com**

**HostName github.com**

**User git**

**PreferredAuthentications publickey**

**IdentityFile ~/<replace-your-file-name>**

**git init --> intialises a folder as git repo**

**git status --> we can know what is the status of our files**

**git add -->**

**100 scripts**

**signup, login, forgot password**

**temp area --> select what is completed and push to git**

**signup --> add to it temp area and push to git**

**git config --global user.name "your-user-name"**

**git config --global user.email "your-email-id"**

**git commit -m "as part of bug fix" --> till here local...**

**signup for github**

**create or use existing pub/private key**

**place pub key in github**

**keep private key**

**create .ssh folder in home directory**

**create config file without extensions in .ssh folder**

**make sure you give your username and email before commit.**

**Host github.com**

**HostName github.com**

**User git**

**PreferredAuthentications publickey**

**IdentityFile ~/<replace-your-file-name>**

**1. signup for github**

**2. we need to generate keys or use existing keys**

**3. keep public key in github**

**4. create .ssh folder in user directory in not exist.**

**/c/Users/user/.ssh**

**5. create config file in .ssh, without extension**

**6. keep the github config**

**/Users/<user-name>/.ssh --> Mac**

**/home/<user-name>/.ssh --> ubuntu**

**Shell Scripting**

**Install MySQL through shell script**

**1. check user is root or not**

**2. if root --> if id is not equal to 0, then it is not root user**

**proceed**

**if not root**

**stop and say run with root access**

**3. now install mysql**

**4. check installed properly.**

**5. if success**

**then good**

**not success**

**show what is the error**

**DRY --> don't repeat yourself**

**VARIABLE\_NAME=VALUE**

**PERSON1=Ramesh**

**PERSON2=Suresh**

**change in a single location will effect everywhere automatically**

**how do you run commands in shell script and take output**

**$(<command>)**

**arguments == args**

**sh 02-variables.sh Robert John Ram Raheem**

**$1=Robert**

**$2=John**

**Shell script disadvantage:**

**100 statements**

**10th line --> error**

**1. stop, clear the error and proceed**

**2. dont worry about error, proceed**

**shell script wont stop if it faces error,**

**it is our responsibility to check and proceed**

**$? --> if success, it has 0**

**if failure, not zero**

**Functions:-**

**less lines of code is preferred doing the same task**

**function --> code that is repeated again, you can keep in function and call it whenever you want....**

**manager --> developer**

**FUNCTION\_NAME(){**

**statements to run**

**}**

**VALIDATE(){**

**statements to run**

**}**

**FUNCTION\_NAME --> calling name**

**Redirections:-**

**command > temp.log --> by default success output only stores here**

**1 --> success**

**2 --> failure**

**& --> both success and failure**

**> --> replace previous output**

**$? --> exit status of previous command**

**$0 --> you will get script name**

**$1**

**$2**

**$N**

**$@ --all args**

**$# -- no of args**

**RED -- \e[31m**

**GREEN -- \e[32m**

**YELLOW -- \e[33m**

**normal -- \e[0m**

**echo -e "Hello Iam learning \e[31m Shell script"**

**Loops:-**

**normal -- \e[0m**

**using loops install any no of packages**

**--------------------------------------**

**sh 13-install-packages.sh git mysql gcc postfix net-tools**

**$# --> no of args**

**$@ --> all args**

**1. check root user or not**

**2. if root user**

**check package is already installed or not**

**if installed skip it and tell user, already installed**

**if not installe, install it**

**validate it**

**if not root user**

**throw the error**

**1. delete old log files, disk usage monitor**

**SED editor = streamline editor = temporary editor**

**--------------------------------**

**CRUD**

**sed -e '1 a I am learning Shell script' <file-name>**

**-e = temp change**

**-i = perm change**

**sed -e '1 i I am learning Shell script' <file-name>**

**lines update**

**--------------**

**sed -e 's/word-to-find/word-to-replace/' --> by default first occurence in every lines**

**sed -e 's/word-to-find/word-to-replace/g'**

**sed -e '/learning/ d' passwd**

**sed -e '1d' passwd**

**Old Log file (16 note):-**

**1. we have a folder where we are storing log files**

**/tmp/shell-script-logs**

**2. delete log files more than 14 days, only .log extensions not anyother files**

**source-directory**

**search .log files and more than 14 days old**

**rm -rf**

**find . -type f -mtime +14**

**| --> or**

**& --> and**

**echo "This is a test mail & Date $(date)" | mail -s "message"** [**info@joindevops.com**](mailto:info@joindevops.com)

**1. user has to provide source directory**

**2. action --> archieve/delete**

**3. if he selects archieve --> where is the destination directory**

**4. time --> optional, if he gives take it, otherwise 14 days default**

**5. memory --> optional. if he dont give dont consider, if he gives consider it...**

**old-logs.sh -s <source-dir> -a <archieve/delete> -d <destination-dir> -t <no-days> -m <memory-in-mb>**

**-s, -a, -d --> check all these inputs, if he dont give tell him the usage....**

**source directory exists or not**

**destination directory exists or not**

**-a --> archieve if he dont give destination dir throw error about destination-dir**

**AWS instances, route 53 records**

**AWS instances, route 53 records (17 note)**

**------------------------------------------**

**1. we need to create ec2 instances**

**2. mongodb, mysql, shipping we are creating t3.small remaining t2.micro**

**3. creating route53 records, web public ip remaining private ip**

**IAM --> identity access management**

**-----------------------------------**

**Authentication --> username/password**

**Authorization**

**Authorization --> you need to have access to enter project bays**

**Roles --> permissions**

**---------------**

**Team Manager --> super admin**

**Team Lead --> admin**

**Senior Engineers --> normal access**

**Trainee --> READ access**

**user == person --> username/password = authentication**

**Authorization**

**----------------**

**Role and permissions**

**user --> what is the role of user ? --> what are the permissions attached to that role**

**Permissions**

**-----------------**

**Nouns and Verbs**

**AWS --> EC2, VPC, Route53, CDN, IAM == AWS resources**

**Resource**

**-------------**

**web, cart, catalouge, hostedzone == nouns**

**create, update, read, delete == actions == verbs**

**sivakumar == trainee**

**-------------------**

**EC2 --> Web --> READ**

**TRISHANT == Junior DevOps Engineer**

**EC2 --> WEB --> READ and UPDATE**

**Sandhya == Senior Engineer**

**EC2 --> WEB --> Create, READ, UPDATE**

**Aditya == TEAM Lead**

**EC2 --> WEB, CART, CATALOGUE, etc --> Create, READ, UPDATE**

**Akhila --> TEAM Manager**

**EC2 --> WEB, CART, CATALOGUE, etc --> Create, READ, UPDATE and DELETE**

**Resources also should have access to access another resources**

**----------------------------------------------------------**

**Roles to resources**

**instance can create other instance**

**aws ec2 run-instances --image-id ami-03265a0778a880afb --instance-type t2.micro --security-group-ids sg-087e7afb3a936fce7**

**ami-03265a0778a880afb**

**1. create one user and gave admin access**

**2. create access keys**

**AKIAUSW45M2WHLZYIMFP**

**fOOpKufyvMMf7ZlyqnmnwGvg1kC+eUB9GRBCYTav**

**created role**

**attached permissions**

**named it**

**attach this role to instance...**

**algorithm**

**---------------**

**1. instance creation**

**aws ec2 run-instances --image-id ami-03265a0778a880afb --instance-type t2.micro --security-group-ids sg-087e7afb3a936fce7 --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=production}]'**

**2. mongodb, mysql, shipping t3.small otherwise t2.micro**

**3. we need ip address to create route53 record.**

**aws route53 change-resource-record-sets \**

**--hosted-zone-id 1234567890ABC \**

**--change-batch '**

**{**

**"Comment": "Testing creating a record set"**

**,"Changes": [{**

**"Action" : "CREATE"**

**,"ResourceRecordSet" : {**

**"Name" : "'" $ENV "'.company.com"**

**,"Type" : "A"**

**,"TTL" : 1**

**,"ResourceRecords" : [{**

**"Value" : ""**

**}]**

**}**

**}]**

**}**

**'**

**improvements**

**---------------**

**if web, get public ip and create records**

**check if records already exists**

**if exists update**

**if not exist create**