1)I have written a script to check disk memory and send a email notification, saying that memory is 90%, and please take appropriate action.

```
Vim memch.sh

#!/bin/bash

Mem=`df –h . | tail -1 | awk –F — — ‗{print $4}‘ | sed ‗s/%//g'` If [ $mem –gt 40 ]

Mail   –s   —memory  reached‖  –c  pradeep@gmail.com   < filename

Fi
```

2)Collect all the logs and deletes files older than X days

```
#!/bin/bash

prev_count=0

fpath=/var/log/app/app_log.*

find $fpath -type d -mtime +10  -exec ls -ltrh {} | rm -rf

count=$(cat /tmp/folder.out | wc -l)

if [ "$prev_count" -lt "$count" ] ; then

MESSAGE="/tmp/file1.out"

TO="daygeek@gmail.com"

echo "Application log folders are deleted older than 15 days" >> $MESSAGE

fi
```

3)REALISTIC SCRIPT 2 – Service installation

```
Services=”ser1 serv2 serv3          —

For service in $services

Do

                        Ps -ef | grep “$service”

                        If [$? -ne 0]

Then
```

```
            Mail -s “service is stopped and trying to restart
Sudo service $service restart
Fi
Done
```

ALL JENKINS PIPELINES

1)BASIC SKELETON OF JENKINS FILE

```
pipeline {
   agent { label 'slave2' }
   stages {
       stage('Build') {
          steps {
                 git branch: 'master', url: 'https://github.com/jenkins-docs/simple-java-maven-app.git'
                 sh '''
                    #!/bin/bash
                    pwd
                    ls
                    echo "This is a BUILD stage"
                    mvn clean package
                 '''
          }
       }
   }
}
```

2) JENKINS PARALLEL JOB

```
pipeline {
  agent none
  stages {
    stage('BUILD') {
      agent any
      steps {
        sh '''
          #!/bin/bash
          pwd
          ls
          echo "This is a BUILD stage"
          sleep 5
        '''
      }
    }
    stage('DEPLOY') {
      agent { label 'slave1' }
      steps {
        echo "This is a DEPLOY stage"
        sh 'sleep 5'
      }
    }
    stage('TESTING') {
      parallel {
        stage('TESTING1') {
          agent { label 'slave2' }
          steps {
            sh 'echo "This is a TESTING1 stage"'
```

```
                sh 'sleep 5'
            }
        }
        stage('TESTING2') {
            agent { label 'master' }
            steps {
                sh '''
                    echo "This is a TESTING2 stage"
                    sleep 5
                '''
            }
        }

    }
  }
}
```

3) JENKINS ENV

```
pipeline {
  agent none

  environment {
    BRANCH = 'main'
    GIT_REPO = 'https://github.com/jaintpharsha/devops-june-2022.git'
  }

  stages {
    stage('Build') {
            agent any
            environment {
                BUILD_ENV = 'only_for_build_stage'
            }
            steps {
                git branch: env.BRANCH, url: env.GIT_REPO
                sh '''
                    #!/bin/bash
                    pwd
                    ls
                    echo "This is a BUILD stage"
                    sleep 5
                '''
            }
        }

        stage('DEPLOY') {
            agent { label 'slave1' }
            steps {
                echo "This is a DEPLOY stage"
                sh 'sleep 5'
                echo "$BRANCH $GIT_REPO"
                echo "$BUILD_ENV"
            }
        }
    }
}
```

4) JENKINS CATCH ERROR

```
pipeline {
   agent any
   stages {
      stage('Build') {
         steps {
            catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {
               sh '''
                  #!/bin/bash
                  pwd
                  ls
                  echo "This is a BUILD stage"
                  sleep 5
                  exit 1
               '''
            }
         }
      }

      stage('DEPLOY') {
         steps {
            echo "This is a DEPLOY stage"
            sh 'sleep 5'
         }
      }
   }
}
```

1) DOCKER BASIC SKELETON FOR JAVA - MAVEN

```
FROM maven
```

```
WORKDIR /app
COPY . /app
RUN mvn --version
RUN mvn clean test
```

2)DOCKER MULTI STAGE BUILD

```
FROM maven AS BUILD
WORKDIR /app
COPY . /app
COPY ./libraries /root/.m2
RUN cd ./target; ls
RUN mvn clean package -Dmaven.test.skip=true

FROM tomcat:8.0-alpine
RUN rm -rf /usr/local/tomcat/webapps/ROOT
COPY --from=BUILD /app/target/calculator.war /usr/local/tomcat/webapps/ROOT.war
EXPOSE 8080
CMD ["catalina.sh","run"]
```

3) DOCKER FILE FOR PYTHON

```
FROM python
WORKDIR /app
COPY ip_app.py /app
COPY requirements.txt /app
RUN pip install -r requirements.txt
ENTRYPOINT ["python","ip_app.py"]
```

KUBERNETES

1) BASIC SKELETON YML FILE

```
apiVersion: v1
kind: Pod
metadata:
   name: my-nginx
spec:
   containers:
      - name: nginx-container
        image: nginx
        ports:
          - containerPort: 80
```

2)DEPLOYMENT.YML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 4
```

```
 selector:
  matchLabels:
   app: template-nginx
 template:
  metadata:
   labels:
    app: template-nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

3) DEAMONSET

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: nginx-daemonset
spec:
 selector:
  matchLabels:
    app: daemonset-nginx
 template:
  metadata:
   labels:
    app: daemonset-nginx
  spec:
   containers:
    - name: nginx
```

```
     image: nginx:1.14.2
     ports:
    - containerPort: 80
```

4)PV

```
apiVersion: v1
kind: PersistentVolume
metadata:
    name: my-pv
    labels:
       volume: test
spec:
    storageClassName: local
    accessModes:
       - ReadWriteOnce
    capacity:
       storage: 2Gi
    hostPath:
       path: "/home/ubuntu/my-pv-volume"
```

5) PVC

```
apiVersion: v1
kind: Pod
metadata:
    name: my-pvc-pod
spec:
  volumes:
    - name: pvc-volume
       persistentVolumeClaim:
            claimName: my-pvc # This name should be same the PVC object name
    containers:
      - name: my-nginx
        image: nginx:latest
        ports:
          - containerPort: 80
        volumeMounts:
        - mountPath: "/usr/share/nginx/html"
            name: pvc-volume # This name should be same as the above volume name
```

6) CLUSTER IP

```
apiVersion: v1
  kind: Service
  metadata:
     name: my-svc
  spec:
    type: ClusterIP
    selector:
      app: my-nginx
    ports:
      - name: http
        port: 30080
        targetPort: 8080
```

7) NODE PORT

```
apiVersion: v1
        kind: Service
        metadata:
           name: my-svc
        spec:
```

```
        type: NodePort
        selector:
            app: my-nginx
        ports:
            - name: http
              nodePort:30082
              port: 8080
              targetPort: 80
```

8) HEADLESS SERVICE

```
apiVersion: v1
kind: Service
metadata:
  name: my-headless-svc
spec:
  clusterIP: None
  selector:
    app: my-nginx
  ports:
   - name: http
    port: 30080
    targetPort: 8080
```

9) INGRESS

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: fruits-ingress
spec:
```

```yaml
ingressClassName: nginx
rules:
- host: test.com
  http:
    paths:
    - path: /orange
      pathType: Prefix
      backend:
        service:
          name: orange-service
          port:
            number: 5678
    - path: /apple
      pathType: Prefix
      backend:
        service:
          name: apple-service
          port:
            number: 8090
    - path: /fe
      pathType: Prefix
      backend:
        service:
          name: fe-svc
          port:
            number: 8011
    - path: /home
      pathType: Prefix
      backend:
        service:
          name: home-svc
```

```
            port:
              number: 8010
```

10) CONFIFG MAP

```
apiVersion: v1
kind: ConfigMap
 metadata:
    name: test-configmap
  data:
  environment: test
   app: frontend
```

11) SECRETS

```
apiVersion: v1
 kind: Secret
  metadata:
    name: test-secret
 data:
  dburl: <base64_value>
   dbpassword: <base64_value1>
```

12) ROLES

Create a Role

```
                              apiVersion: rbac.authorization.k8s.io/v1
                              kind: Role
                              metadata:
```

```
  namepsace: default
  name: test-role
rules:
        - apiGroups: [""]
          resources: ["pods"]
          verbs: ["get", "list"]
```

ClusterRole

- this is cluster wide role

Create a ClusterRole

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: test-cluster-role
rules:
        - apiGroups: [""]
          resources: ["pods"]
          verbs: ["get", "watch", "list"]
```

12) ROLE BINDING

RoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  namespace: default
  name: read-pods
subjects:
- kind: ServiceAccount
  name: test
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: test-role
```

ClusterRoleBiniding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  namespace: default
  name: read-pods
subjects:
- kind: ServiceAccount
  name: test
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: test-role
```

13) TERRAFORM FILE FOR EC2 CREATION

```
resource "aws_instance" "ec2_provisioner" {
  ami = var.ec2_ami
  instance_type = var.ec2_instance_type
  key_name = var.ec2_pem
  vpc_security_group_ids = ["${aws_security_group.ec2_sg.id}"]
  tags = {
    Name = "PROVISIONER"
  }
}

resource "aws_security_group" "ec2_sg" {
  name = "ec2_sg"
  description = "Allow ssh and http"

  ingress {
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port = 80
    to_port = 80
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

14) TERRFORM FILE FOR S3 BUCKET CREATION

```
resource "aws_s3_bucket" "s3_backend" {
   bucket = var.bucket_name
}

resource "aws_s3_bucket_versioning" "versioning_backend" {
  bucket = aws_s3_bucket.s3_backend.bucket
  versioning_configuration {
    status = "Enabled"
  }
}

resource "aws_s3_bucket_server_side_encryption_configuration" "s3_backend_sse" {
  bucket = aws_s3_bucket.s3_backend.bucket
  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm = "AES256"
    }
  }
}
```

15) TERRFORM FILE FOR DYNAMODB

```
resource "aws_dynamodb_table" "s3_backend_locking" {
    name = var.dynamo_name
    hash_key = var.hash_key
    billing_mode = var.bill_mode
    attribute {
        name = var.hash_key
        type = "S"
    }
}
```

16) ANSIBLE PLAYBOOKS

Playbook to install nginx on RHEL distribution

```
tasks:
 -name:create a directory nginx
  file:
    path: /ubuntu/home/nginx
    state: directory

 -name: Installing nginx
  yum:
    name:nginx
    state:latest

 -name: start nginx
  service
    name:nginx
    state:start
```

Playbook to install tomcat on ubuntu

```
tasks:
 -name: installing tomcat
  apt:
```

```
  name: tomcat
  state: latest
-name: start tomcat
 systemctl:
   name:tomcat
   state:start

-name: status of tomcat running or not
 systemctl:
   name:tomcat
   state:status
```

Playbook to install tomcat on server using groupname from inventory

```
-name:play_1
 hosts: frontend
 tasks:
  -name: install tomcat
   apt:
    name: tomcat
    state: latest
  -name: start tomcat
   systemctl:
    name:tomcat
    state:start

- name: Play_1
  hosts: backend
  tasks:
    - name: Install git (backend)
      become: true
      apt:
        name: git
        state: present
        update_cache: yes
    - name: Install jq (backend)
      become: true
      apt:
        name: jq
        state: latest
        update_cache: yes
```

Installing multiple items with single task

```
- name: Demo
  hosts: frontend
  tasks:
    - name: Install git (frontend)
      become: true
      apt:
        name: "{{ item }}"
        state: present
        update_cache: yes
      with_items:
          - git
          - nginx
          - memcached
          - jq
          - curl
          - wget
```