**Part 1: Ansible and basic Linux concepts**

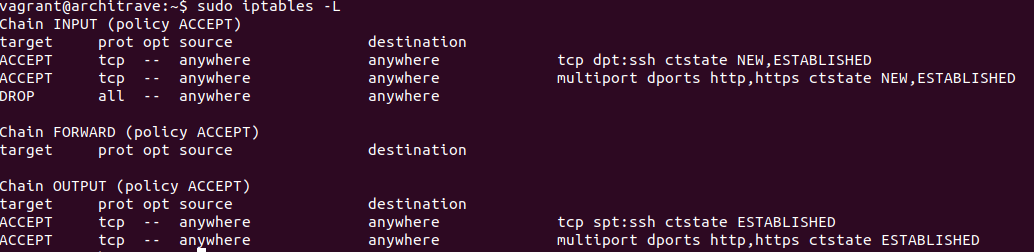
1. Provisioning of the playbook is broken, you should fix it in order to move forward when everything is working you should get Ìnstance provisioned successfully message. Take into account all aspects of the code

* Issues encounter was the role DB handlers main.yml was in lower case and in the principal yml was called with the first letter in upper case (backup of the original file has been done and a comment provided). Later than that we were able to finish the ansible run without issues.

Note: For this point I create a video that shows from the beginning to the end of the vagrant configuration with the ansible setup happen. <https://www.amazon.com/clouddrive/share/CuSSYeBIjCvs9XpLROs7Gs1QUnt8rCHbe1ltQX1BDem>

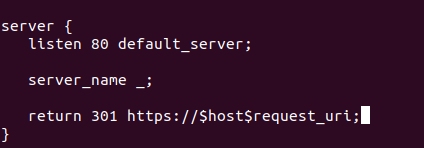
1. The system as it is allows all traffic to the server, create a role to limit access to the server via ports 80, 443 & 22 using iptables.

* Iptables have been created as requested. Screenshot below



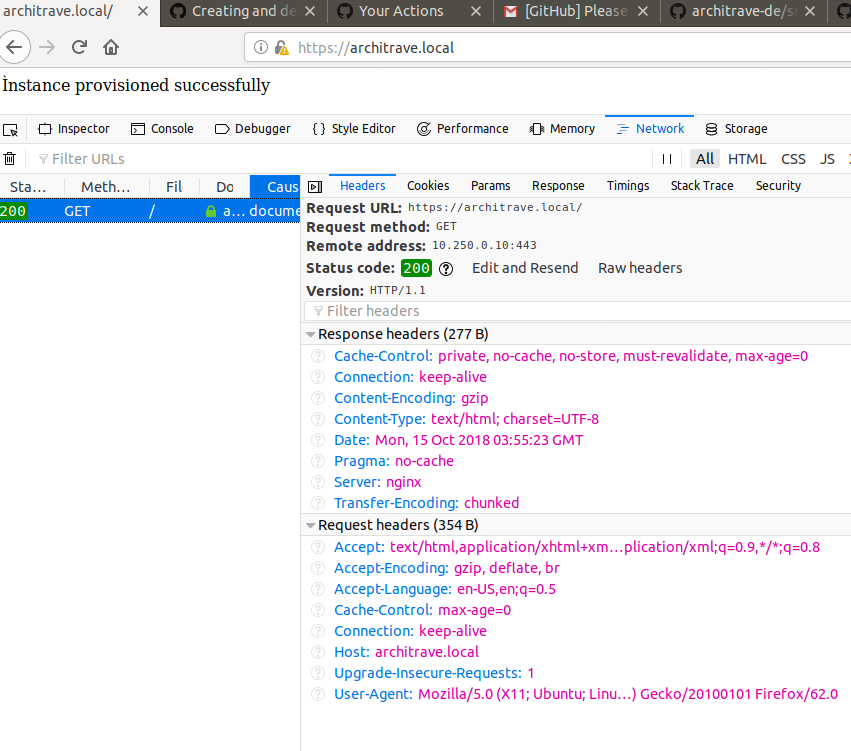
1. Add http redirection so all traffic will be served via https

* Redirection done via changing the http nginx file architrave.vhost.conf



1. Make sure the client is not receiving the Nginx version as part of the http headers

* Nginx.conf file change the server\_tokens off;



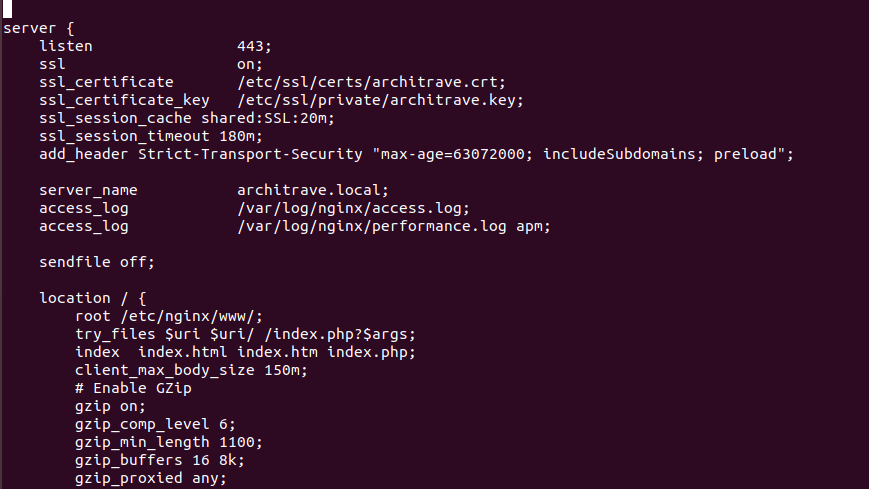
1. Create user testuser with password qweRTZ123

* User created with the password required.



1. Change the location of the website files to a more secure folder and grent testuser permissions to read the folder content

* Location have been changed as requested and no permissions changed as currently the permissions are 644 and user can read the file without issues.



**Part 2: Docker**

1. Create a new folder docker at the root of the repository

Folder created.

1. Migrate the code to docker using micro-services architecture, all the files should be stored inside the docker folder.

Dockerfile created as requested and as well a sh script to automate the DB management. Have in mind the creation of the Docker file was following the ansible step by step.

**Part 3: Questions**

* What elements are missing to make this a 'production' ready system? Please list them in order of priority and add a short explanation.

**Part 4: Kubernetes**

1. What steps are needed to run the system on kubernetes? Please add a concise description of involved steps. If time allows, start implementing it.

I will merge part 3 and part 4 as I think is the correct technology to use for these applications and all kind of applications that don’t have a hardware requirement to be physical. In this step by step we contemplate the scenario where we use AWS (Cloud) and microservices. For the orchestration we can use Kubernetese and will be good to use Rancher as the orchestration of the k8s

* Get the customer requirements and user stories. This is important because in this way we can understand the customer needs. We should focus on customer as they will be the end users and depends of those needs the infrastructure will change.
* Understand the application needs and dependencies. We need to understand the dependencies like web, db, storage, ports and general security.
* Create the docker image. Depends if the vendor has their own image you just need to request them but in case the vendor don’t have their own image we need to create them from scratch. All in Docker and Kubernetese is managed as “Infrastructure as a code” so all the process need to be capture in the docker file.
* Upload the image into a docker hub. We need to have the image available in a repository to be download in the moment that is needed.
* Now is the moment to request the infrastructure for the Kubernetese (in case we don’t want to use the EKS). We need at less 7 VM’s (1 Rancher, 3 master’s and 3 Workers). Can add more works as needed.
  1. Create the VPC, internet gateway and subnets. (This part is important as contemplate the security with the ACLS and Security Groups)
  2. Request the EC2 (VM’s). Will be good have already an image with all the customizations needed. In case not available we can have an ansible as well for the configuration manager.
  3. Storage. We need to request a NFS that will be shared between the works and this allow us to have stateful applications. Is recommended to request dedicated storage for performance NETAPP.
  4. DB’s. Databases is a good option to be sent to RDS (Database as a Service) so not maintenance is needed by the team. In some cases, if the DB’s can’t be stored in RDS we can create stateful-sets pods to host the DB’s.
  5. Elastic Load Balancers.
* Ones you have the Infrastructure ready to implement we need to determine what Kubernetese we are going to implement
  1. KOPS – For Cloud K8S
  2. Minikube – For Onprem
  3. Rancher. Is the best option as you can manage multiple k8s clusters at the same time.
* Once you have the Rancher installed and all nodes ready is time to implement the application. All the implementation can be done by yml files and this are some of the services needed for the application.
  1. Persistence Volume: The PV is the NFS that is mounted on the workers.
  2. Persistence Volume Claim: Each PVC contains a spec and status, which is the specification and status of the claim.
  3. Deployment: in the deployment you specify the container settings.
     1. Name: Name of the deployment
     2. Replicas: How many pods simultaneous.
     3. Containers:
        1. Image name: Image that will be download. You should specify if is dockerhub (default) or another provider. Credentials should be passed as secrets.
        2. LivenessProbe: This setting will in charge of the self-healing. If detects a pod is unavailable will recreate.
        3. Ports: The ports that will listen. Is important to note that pods will have a port for communication internal of the cluster and another port for external facing.
        4. Volumes: The Volumes for PV and PVC.
        5. Secrets: The credentials for the hub if the image is private.
  4. Service: The service will act as a DNS and as well depends of the configuration can expose the ports externally.
  5. Ingress: The ingress is a reverse proxy with NGINX and will allow to hear multiple websites to the same port in the cluster via labels in the URL.
  6. Load Balancer: This is not a Rancher service, but we can setup a container to hear on port 31485 and via load balancer perform the port forwarding to 80. This is the easy way to expose a website in conjunction with the service.

All the above services can be implemented via yml files. All in Kubernetese can be done via API and yml files. To manage Kubernetese via CLI you need to install first kubctl and configure the connection to the server.

* The above steps are to implement maybe first in Pre-Production and follow the path to production. The advantage of use Kubernetese is that you don’t need infrastructure for every single step in the path to production as you can handle via projects and namespaces.
* Once the test is done on Pre-Production namespace you can proceed with the migration of the same image to production. Is really easy as well to perform a rollback.