Features seem to be fairly equal between Gitlab and Azure, so I don't think either really wins out by features.

https://about.gitlab.com/devops-tools/azure-devops-vs-gitlab.html

With Github / Gitlab, however TRI-AD would be able to provide support for hosting / maintenance. There is also (we think) more general familiarity with these tools, which are simpler to use and would make it easier for new or outside collaborators to start contributing.

Options:

Really depends on what features we require:

https://about.gitlab.com/pricing/#gitlab-com

https://github.com/pricing
https://github.com/organizations/new

Github pricing in general is better (however this of course depends on the tier we would require).

TRI-AD would be able to provide support for the following:

- Github host (self-hosted or public) + Gitlab CI Runners
  - Public Github + Gitlab CI Runners seem to be the best option
  - Maintain public repos, Gitlab runners are simple, easy to use
- Gitlab host (self-hosted or public) + Gitlab CI Runners
- Github host (self-hosted or public) + Jenkins Runners (we use this setup internally right now, so this would probably be easiest to support although that means we are "stuck" with Jenkins

Support we could provide:

- Deploy and update stack to a provided AWS account (we have Terraform recipes for these already which would only require minor modification)
- Maintain Gitlab / Github / Jenkins stack

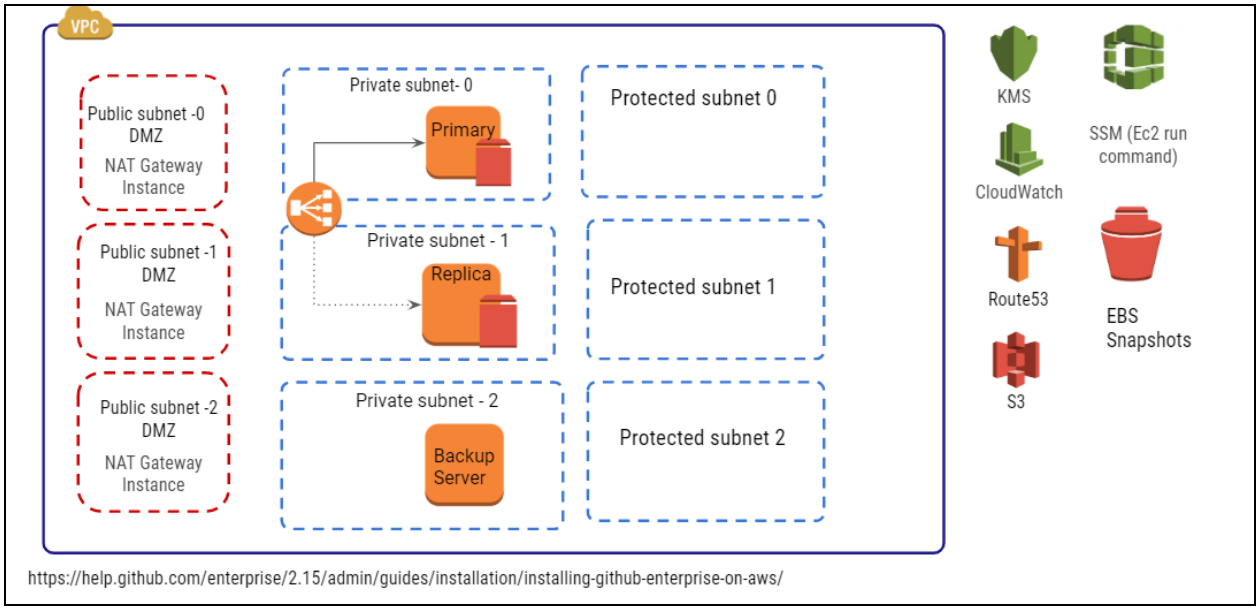
Some questions which would probably help guide us towards the right choice:

- Number of users?
- What features do we need (guests? Collaborators? Issue tracking?)?

- All open source projects or a mix?

Architecture of TRI-AD Github Enterprise

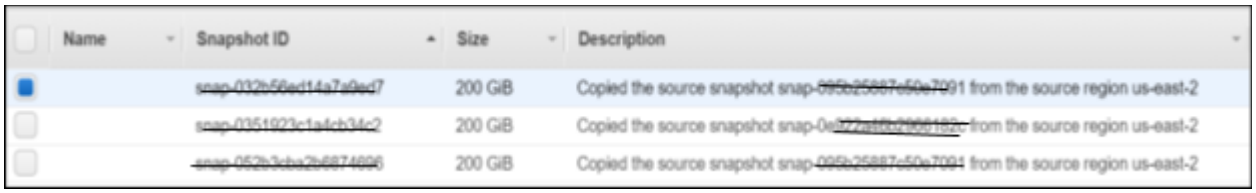
- Primary and replica (HA)



How is GHE backed up?

There is a data lifecycle management policy, which takes periodic snapshot of the GHE primary instance's data volume by searching for its unique tag name. The policy runs for every 12 hours and stores the snapshot in the same region.

Then, we have a lambda function running on the primary region and it gets triggered by a CloudWatch event every 12 hours. This lambda function gets the list of all the snapshots created by DLM, fetches the latest snapshot by filtering on the creation date, and then copies them to the DR region for redundant storage.

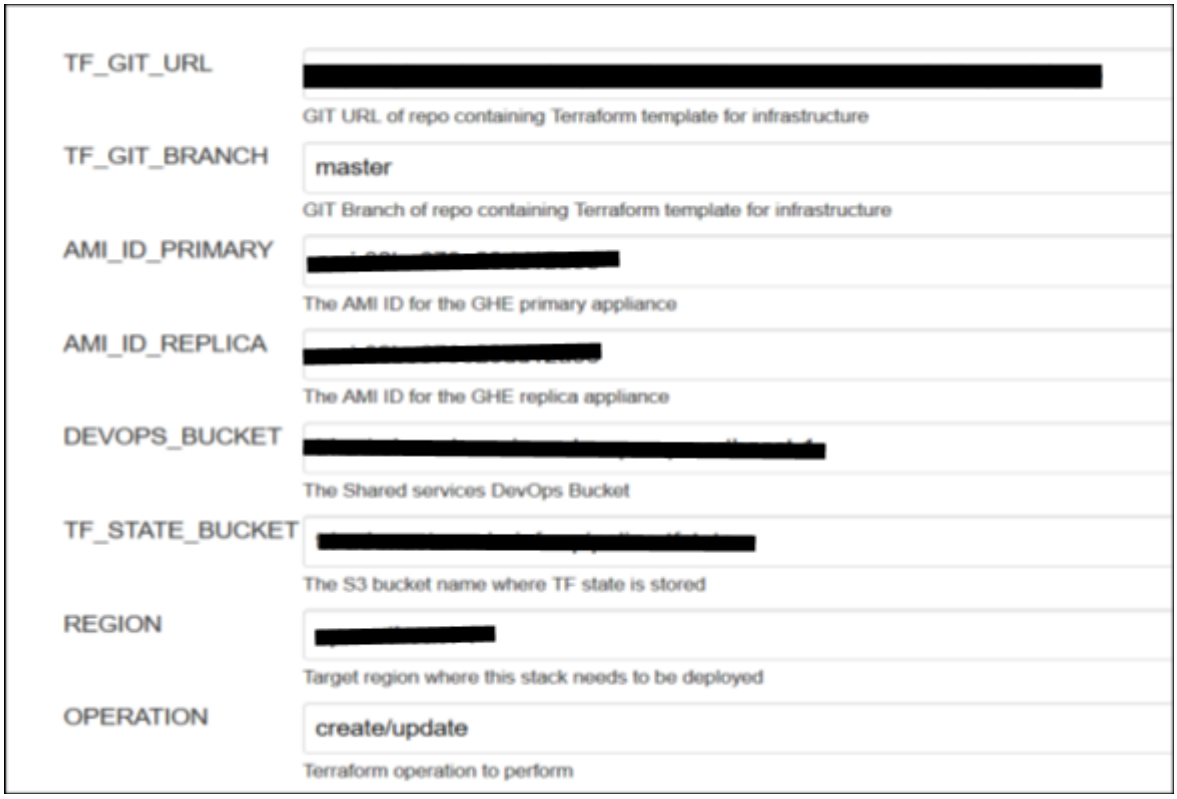


How is GHE deployed?

The pipeline job, **deploy-github-enterprise** deploys all the required application infrastructure resources into the target environment. The pipeline gets parameters for assuming a role into the child account from the master account, and then performs a terraform deployment to create the required resources.

Following are the parameters required to execute the pipeline for deploying GHE Appliances:

The pipeline job, **deploy-github-enterprise** deploys all the required application infrastructure resources into the target environment. The pipeline gets parameters for assuming a role into the child account from the master account, and then performs a terraform deployment to create the required resources.



How is GHE upgraded?

On a high level, the upgrade pipeline does the following:

- Triggers the "**bake-ami**" pipeline with the latest GHE AMI ID to bake an AMI with all the pre-configurations
- Runs a SSM Automation document which puts the GHE primary instance in maintenance mode, and removes replication configuration from the replica instance
- Triggers the "**deploy-github-enterprise**" pipeline to replace the primary instance with the new AMI
- Triggers the same pipeline again to replace the replica instance with the new AMI

The Upgrade process

1. Baking of new AMI:

- When a new GHE AMI is released, the "**bake-ami**" pipeline is used to create an encrypted AMI with all the required tools like SSM agent and CW agent
- Refer section 26.4 to know more what the AMI baking process for GHE

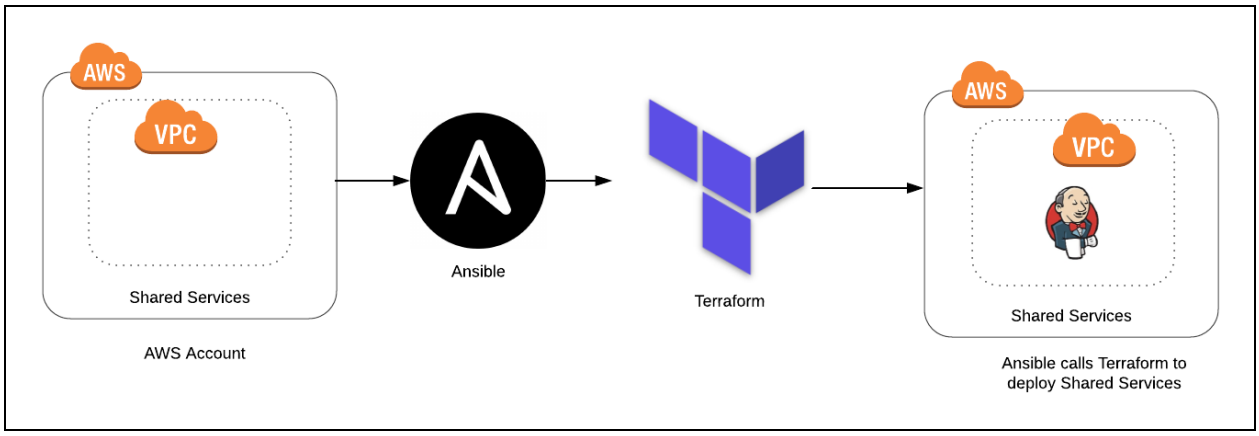
2. SSM Automation Document:

- SSM automation runs workflows to perform common maintenance and deployment tasks if EC2 instances and other resources.
- When this document is executed from the upgrade pipeline, it does the following:
    - Puts the GHE primary instance in Maintenance mode. This is to prevent GHE from being online for pushing code.
    - Next it stops replication on the replica instance, this is done to maintain consistency with primary node when the EBS volume is re-attached to the new instance.

3. Upgrade of the Primary/Replica instance:

- After successfully executing the SSM document, the upgrade pipeline does a Terraform deployment by triggering "**deploy-github-enterprise**". It dynamically provides the new AMI ID for GHE primary instance.
- During deployment, the EBS data volume gets detached from old primary and attached to the upgraded primary instance.
- Once deployed, the pipeline waits for the Primary instance to be healthy, by CURLing the health check endpoint periodically.

TRi-AD Jenkins Setup



The high-level process to deploy the Shared Services (shared account in AWS in TRI-AD hosting shared devtools) is:

- Clone the Shared Services repository
- Configure the variables for Terraform and Ansible
- Run the Ansible Playbook. It will:
    - Checkout the Cloudformation Modules
    - Deploy the Shared Services Environment
    - Jenkins created in AWS environment

How are Jenkins Agents Setup?

The shared services **Jenkins Master** is responsible for deploying the required resources to all the AWS accounts. As a best practice, it is recommended not to overload the Jenkins Master by running jobs directly on the master itself. Instead, **Jenkins agents** are created using the Jenkins EC2 plugin to dynamically spin-up when any job is scheduled.

- The Jenkins agents have the same capabilities as the Jenkins Master. They also use the same AMI and security group.
- The SSH key for Jenkins agents is generated and uploaded to the EC2 key pairs. The EC2 plugin uses the newly generated key pair to spin-up the Jenkins agents.
- The cloud-init configuration within the EC2 plugin installs the build dependencies for the Jenkins agent to start. The current values set for the agent instance type is **t2.medium**.