Challenge

You are going to be part of an SRE organization which gives solutions to the vertical product team for their development. As an enterprise standard all customer facing web applications are built on microservices architecture to have better scale of delivery and operation. For the last six months, the vertical product team had a problem with respect to the enterprise scale CI/CD platform, they were moving too slowly to deliver. They had been on the critical path for all new product releases for years, and the inability to scale is affecting further adoption of pipelines. They are looking for an indigenously developed enterprise scale CI platform.

There are three high-level goals for the enterprise grade CI platform.

1. Build a highly scalable, available, fault tolerant, distributed Enterprise CI platform using open source Jenkins on AWS.
2. Provide templatized CI pipelines for one technology of your choice (Java, Node.js, .Net etc.) for the consumption by vertical teams.
3. In-built capabilities to trigger notifications via various channels in case of any failure or anomaly.
4. Dashboard to visualize the health of pipelines and platforms.

Task for Today

You are assigned a task to build an enterprise scale Jenkins based CI platform on AWS cloud which can be consumed by application teams without any constant hassle. The majority of release management activities such as build, static code analysis, testing (unit, functional, integration, load) should be automated and templatized using scripts for easy consumption by teams

While the developers are writing code, you have to enable them to build and test code themselves using CI pipelines with all quality checks in place. They should be able to deploy the services remotely using a combination of IAC formulas and CI/CD pipelines.

Your solution is expected to cover the following deliverables:

1. High-level architecture to demonstrate how you approached the problem and your solution.
2. A GIT repo carrying Infrastructure provisioning templates/files.

3. A GIT repo carrying code for dynamic agent provisioning.
4. A GIT repo carrying pipeline setup using pipeline as a code script.
5. Provisioned infrastructure should be well defined with proper tagging and naming convention.
6. Provisioned infrastructure and deployed services should be secured and have well defined security groups and ACL's to restrict access.
7. URL of Jenkins app which can be accessed in browser.
8. URLs of monitoring portal to visualize the health of the platform.

Don'ts:

- Do not check anything into GitHub with AWS keys.