Project Echo Cloud Plan

Author: Adam Edwards

With contributions on planning from Neel Vishalbhai Lodhiya, cloud team lead.

Overview

This document is an overview of the plan to get Databytes Project Echo running within the public cloud using Google Cloud Platform.

This plan has two primary focuses:

- Outline the foundation to start running the project long term in GCP.
- Provide the architectural design for the first components to be deployed onto GCP, ideally within T2 2024.

The initial plan to begin deployments to GCP has been outlined as the first phase. Subsequent phases for the continued deployment have been more broadly described.

Project Echo Components

- Echo Store
- Echo API
- Echo Engine
- Echo HMI
- Echo MQTT server
- Echo Redis
- Echo Simulator
- Echo IOT Backend

Descriptions of these components are available within the currently existing documentation for Project Echo, except the Project Echo IOT backend is new. It is a MQTT broker configured with authentication that is publicly exposed with the purpose of being used by IOT devices.

Considerations

The primary considerations are:

- This is a university project, so a balance needs to be struck between doing things properly as if this were a professional setting and keeping them simple.
- As the public cloud is being used rather than individual developers' laptops security is of great concern.
- Costs should be kept as low as possible, there is only a very small budget ($500 GCP credit) for Trimester 2 2024.

Security

The Project Echo security team should be involved in the ongoing deployment/running of the services on GCP to ensure that it is secure. How they do this should be for them to implement, but the primary concerns are:

- Never allow anonymous authentication to publicly exposed services.
- Any sensitive data (such as credentials to connect to our services) must be encrypted if it travels across the public internet.
- The credentials used to connect to externally exposed systems (any credentials, really) should not be stored in source code as at least some of Project Echo's Github repositories are public.

Software Configuration Management (SCM)

Software Configuration Management (SCM) relates to how the deployments to the cloud are management. This section provides an outline of how it's currently being managed and can be managed going into the future.

- Environment Management
- Source Code Management
- Build Management

SCM includes many items that largely relate to workload management "how we work" but Project Echo already has processes in place and for now my opinion is to keep things simple, so only the three items above have been elaborated upon.

IE, something that is very important in a professional setting is Configuration Control/Change Management. As the documentation in its current state will show our immediate plan is to first get Project Echo functional within the development environment, as there's no deployments to higher environments Change Management planning isn't required yet.

Wikipedia's page on SCM is a good high-level overview
https://en.wikipedia.org/wiki/Software_configuration_management

Environment Management

A system/application should have multiple environments to support the Devops / Quality Assurance lifecycle and environment management relates to the management of these environments. For Project Echo see the supporting documents Environments Overview and Environments Configuration. The former provides an overview of the environments and how they're used, the second outlines how this can be done for Project Echo.

Source Code Management

Source code management relates to the management of a system's source code and practices that surround it.

Project Echo manages source code using git with repositories stored and shared upon github.com within the Databytes organization. The large single repository is currently in the process of being broken down into one per team.

Project Echo is overall doing quite well with source code management, student leadership being aware that CI and quality of work is important, with care being taken to ensure that main branches are always in a good state.

What would be good to see is an uplift in developer workflows where individual contributor's branches are shorted lived. I recommend leadership learning about trunk based development https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development

Build Management

Build management consists of the management of tools used to prepare code or code artifacts for distribution. What is mostly relates to is CICD tooling (such as Jenkins, Github Actions or Azure Devops), pipelines for CI, build and deployment purposes.

The team is currently somewhat using Github Actions (workflows) for CI purposes in some places. This is good and I suggest that these continue to be maintained and uplifted in ways that provide value. Linting and unit testing are the lowest hanging and most easily implemented CI checks.

There would complexity added and additional overhead on Project Echo if it were to deploy and maintain a CICD tool such as Jenkins. For deploying to the cloud Terraform does enable collaboration via its shared state feature, and cloud engineers can perform build of artifacts (build Docker images, compile Node/React applications, build python packages) on their local machines. So, cloud engineers can perform all of the necessary build and deployment tasks on their local machines whilst maintaining consistency/no drift with source code or Terraform state. https://www.terraform.io/

Phase 1

Phase 1 involves the initial deployment to the development environment/project within GCP, which includes foundation infrastructure to run the Project's components and those components which are core dependencies of others.

This includes:

- Setting up the Project Echo Cloud team for success by having the necessary tools and documentation available
    - A git repository project_echo_cloud to contain the cloud team's code and documentation.
    - Terraform foundations need to be set up and in place within the project_echo_cloud repository, with cloud engineers upskilled in its use
    - Terraform state bucket needs to be deployed and accessible by cloud engineers
    - Access to the GCP development project
    - Other documentation is required to onboard cloud engineers this trimester and in the future. IE, such as getting started with GCP deployments, getting started with Kubernetes, how to build an
- A private docker repository is required to store build docker images. This should be done using a Google Artifact Registry Repository.
- Google Kubernetes Engine Cluster

    - This is the Kubernetes cluster services will run on
- Echo store deployed to the GKE cluster
    - This is the data store used by components.
- Echo API deployed to the GKE cluster
    - This is the interface used to interact with Echo Store.

A document Project Echo GCP Architecture Phase 1 exists which relates to the architecture involved during phase 1.

Phase 2 (and onwards)

Cloud engineers will require Editor access to the development GCP project each trimester and I recommend getting this sorted as early as possible.

This phase and onwards should see cloud engineers upskilling in Kubernetes, Terraform and GCP as soon as possible each trimester.

What is then to be developed and deployed is the remaining Project Echo components. These being:

- Echo Engine
- Echo HMI
- Echo MQTT server
- Echo Redis
- Echo Simulator
- Echo IOT Backend

Dependencies between the components need to be identified, then the dependencies should be deployed before dependent components, so further analysis and planning is required. The other teams involved in Project Echo will need to be involved as components may require additional changes before being deployed (ie, some may point at ports on localhost which will need to change to be variable driven).

There is a proposed architecture for the IOT ingestion backend which has been completed, see the document IOT Ingestion Architecture Proposal.

Addendum (end of T2/2024)

project_echo_cloud repository has been setup with everything necessary to start deploying to the development GCP project, including documentation on getting started with the various technologies.

The GKE cluster, docker repository and other foundational infrastructure is in place.

Echo Store and Echo API component deployment in progress.

The IOT ingestion backend work has been completed except for completing the message processing and push to Echo API code, which is work for the IoT team.

Reference links

https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development

https://www.terraform.io/

https://github.com/DataBytes-Organisation/project_echo_cloud