cloudServe Archetypes

Introduction

A CloudServe Archetype is a set of cloud resources with a predefined deployment strategy and template. These resources may be deployed together as part of the Archetype, or as separate modules, depending on needs.

This document will assume the following:

1. Archetypes are deployed as singular instances and do not deviate from the defined archetype
2. Archetypes may be reused multiple times taking input from user generated content as parameters
3. Archetypes will execute Terraform modules as the Archetype definition, and will not be backported to support other IaC frameworks that may be in use.

Archetype Definition

CloudServe Archetypes are based on the idea of Java Maven Archetypes, or of Stack Sets in the DevOps world. They are simple metadata files which will allow the IaC code to execute a set of predefined modules and scripts to fulfil a business need.

Archetypes must be standardised, as well as extensible. What this means is that the base Archetype will only deploy a very basic subset of resources which may or may not be immediately useful. Subsequent Archetypes may inherit the base Archetype to extend it, and create more complex systems of resources.

Archetypes are defined as JSON or YAML documents. A few keywords will be available to extend Archetypes by inheritance, such as USES or FROM, with the rest of the extended Archetype definition to follow as YAML or JSON definitions.

Archetypes for initial consideration

The following list of Archetypes is the base list and is in no way final.

1. Base Archetype – This is a very basic archetype and contains only the bare necessities to get started. In Azure, this would equate to Vnet and associated peering within a resource group. AWS would encompass a VPC and required network connectivity facilitated by CloudWAN.
2. Clickstream Analytics – Making use of streaming services such as Event Hubs or Kinesis for analysing clickstream data. This may also be analytics on static data which has landed in storage accounts.
3. High Performance files making use of NetApp Files – mainly an Azure based Archetype, depending on the availability of high performance file storage in other CSP's
4. Redis Cache with Tableau Cloud – Archetype to explore data management and reporting
5. Anonymous Credential Service – Archetype for masking log data with any potentially sensitive information

6. Automated Data API using GraphQL – Archetype to deploy an application service to expose an API implementing GraphQL
7. Automated Vision Training – Train a model on ChatGPT-Vision-Turbo and deploy it
8. Streaming Data infrastructure – Consume streaming data
9. Video data Ingestion – Ingest Video and Audio data easily for various functions.
10. AI Performance tuning simulator - simulate the compute, memory, and network performance of AI training clusters
11. Automated Schematization of Data at GM
12. Distribution large object content at Scale - Distribute large, widely -consumed objects (so-called hot content) efficiently to hosts. commonly distributed content types such as executables, code artifacts, AI models

YAML

Archetype definitions to be stored in VCS like GitHub, could be defined as YAML. This would make it easier to maintain, as well as easier to inherit. This is somewhat unrelated to the data model above, as it should simply be a definition for deployment.

Additional Components

All additional components required for successful operation of the chosen Archetype must be packaged along with the resource deployment. These components can be:

- Databricks Cluster definitions
- Databricks notebooks
- Python code
- Java Code
- Function App code
- etc

Terraform Cloud and GitOps

Terraform Cloud must be the sole source for deployments and maintenance of state. All operations must be done through TF Cloud and no teams should be managing infrastructure individually. TF Cloud is also able to manage time bound constraints and this functionality will be leveraged heavily. All archetypes will be a part of the TF Cloud “no-code” solution set. Archetypes may be re-used when deploying to other accounts/subscriptions/projects, but this is not the primary use case.

All Archetype definitions will be released as GM InnerSource in order to be highly extensible and accessible for user collaboration and extension.

To support this, archetypes must be inheritable and extendable.

Potential Issues

- Removal of Resource Groups may be inhibited due to resources deployed by policy