Apigee

Jenkins X

GraphQL

Personas

Cluster Administrators (UBS)

- Responsible for setting up Kubernetes
- Making sure that the worker nodes are in good health
- Responding to any issues with the system in general

Cluster Developer (Adam)

- Keeps track, implements new K8s features, and makes sure the existing cluster adheres to current specifications

Application Administrator (Architect + Team Lead)

- Verifies that deployments and services for applications are working as expected
- Architects the microservices to the Kubernetes environment

Application Developer (Team)

- Creating the application and doing all the things to make sure the microservices work from a code perspective
- Adhering to the integration standards set out by the app admin

(VM) Machine VS Container

Cost and Management Overhead VS **Provision and Automation**

Benefits of Microservices

- Language independence - use right language for the task
- Avoid bottlenecks with scaling
- Deployment benefit
- New Opportunities - containers and serverless paradigms

Prerequisites

- Rapid provisioning
- Rapid Application Deployment
- Basic Monitoring - nice tool to have as the integration between services are complex

Microservice Integration

- Typical integrations: RPC, REST over HTTP, Messagen
- Resiliency Patterns: - at least a retry
    - Circuit breakers,
    - Bulkheads,
    - Fail Fast
- Design Rules:
    - Design service API carefully - could be very painful if it’s not correct
    - Coordinate changes between multiple clients
    - Define dependencies and appropriate versions
    - Client API libraries
    - USE Service Discovery - Ability to search and understand your services - service registry

Microservices patterns - 12 Factors

The twelve-factors app is a methodology that allows for automation, continuous deployment, easy onboarding of new developers, and portability between execution environments. By leveraging this methodology when building modern web-based applications, you can also achieve straightforward deployment on numerous cloud platforms, and high scalability without the need to change tooling, architecture, or how your team works.

What Is Cloud Native?

Factor 1: SCM and revision control (In progress)
Factor 2: Manage dependencies
Factor 3: Application configuration
Factor 4: Backing services
Factor 5: CI/CD
Factor 6: Run processes
Factor 7: Port binding
Factor 8: Scale with processes
Factor 9: Dispose it all
Factor 10: Environment uniformity
Factor 11: Use your logs
Factor 12: Administration

In other words:

1. Codebase must be tracked in version control

2. Dependencies - are explicitly declared and isolated
3. Configuration - never go to app - should be under environment
    a. Config Server
    b. Eureka
    c. Ribbon - software Load Balancing
    d. Feigh and Hystrix
4. Backing services - should be easy to deploy and run
5. Build, Release, Run
6. Processes - execute the application as a stateless process
   Sticky sessions need to revisited and re-implemented
7. Port Binding - expose services via port bindings
8. Principle 8,9 and 10 -
   Concurrency: Scale out with the process model
   Disposability: Quick application startup and shutdown times
   Dev/prod parity : Application is treated the same way in dev, staging, and prod

Principles 11 and 12

Log Management - treated as an event stream
Admin Tasks - treated the same way like the rest of the application

Microservices patterns in Kubernetes

Architecture Grouping

1. Building Blocks

   Code stored in source control - container images
   Push the code -> Build the code and run the tests -> build container image and push the image to repo
   Application modeled in K8s as the Deployment and pods
   Single POD can have many containers inside

2. Deployment Patterns

   Application configuration in kubernetes
   - ConfigMaps: meta data, version
   - Secrets: for sensitive data like password
   Load into POD via environment or as a file
   Tag container image - same as for artifacts
   Groupe containers - Deployments, Replica sets, DaemonSets, OR simply use Helm

   Keep application stateless and do not rely on sticky sessions

3. Runtime Patterns

   Disposability
   Pods managed by replica sets -
   Logging to Elastic is by default works

Spring Cloud Features

❖ Service-oriented architecture
❖ More encapsulated domains but still loosely coupled
❖ Decoupling application and not just a component

Data Microservices
Process microservices
Application microservices

Check Hateoas later to get data as a repository over a proxy

- Config Server
- Eureka
- Ribbon - software Load Balancing
- Feigh and Hystrix

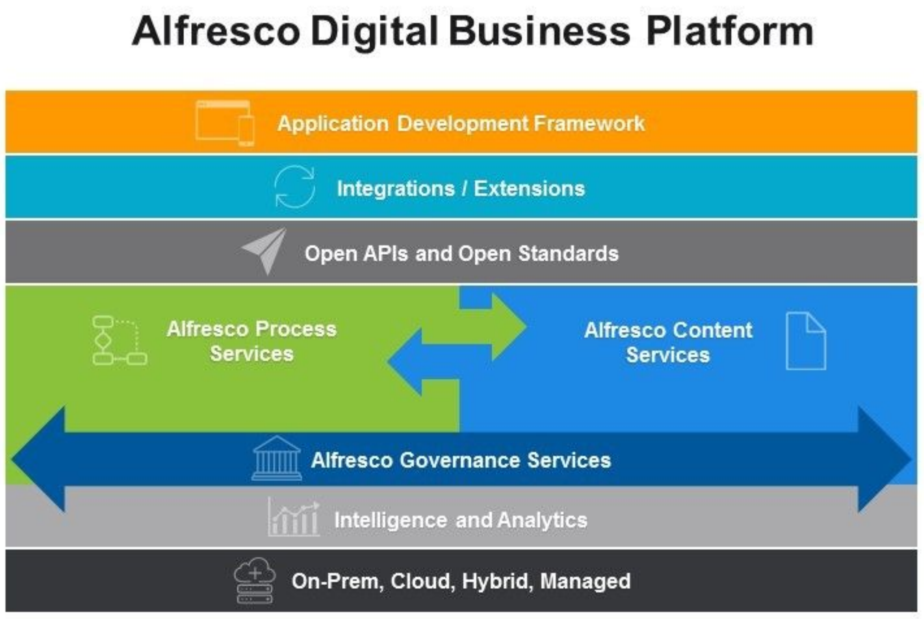
Building a Web Application on Microsoft Azure

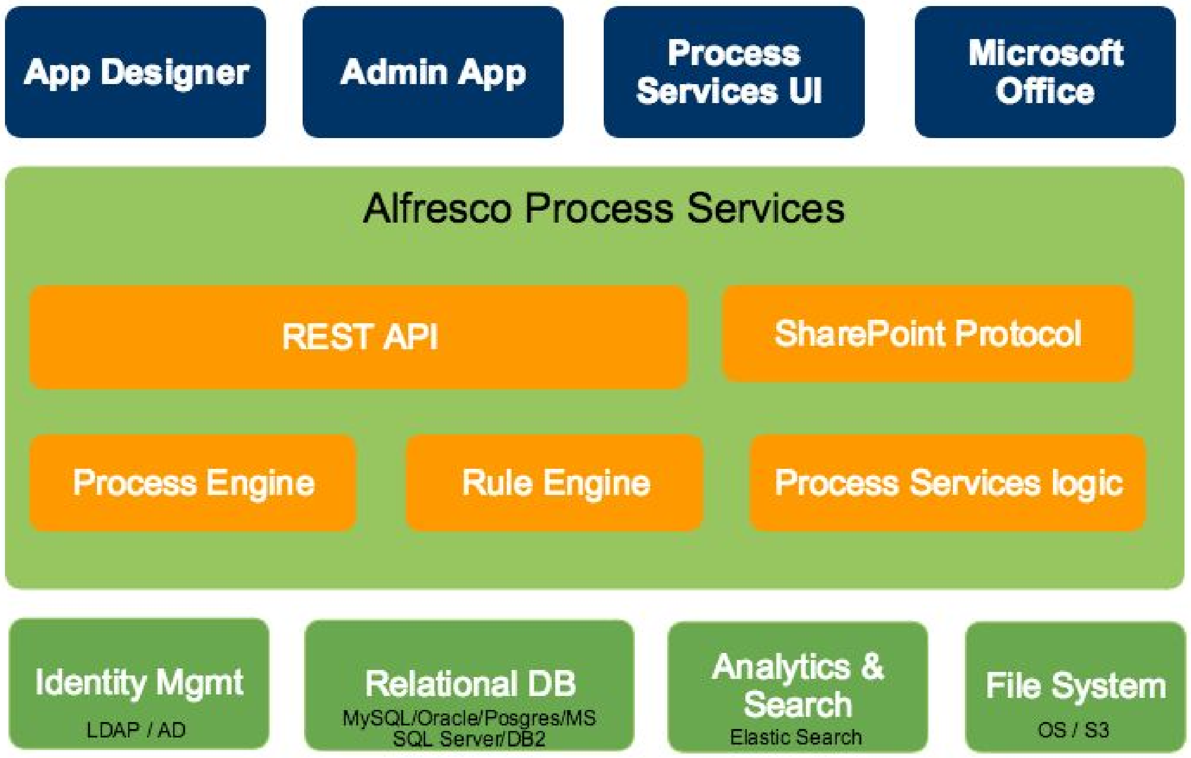
Set the right resources based on rules set
Admin panel -> Web Api -> to call Azure Functions - scale easy
Azure storage explorer

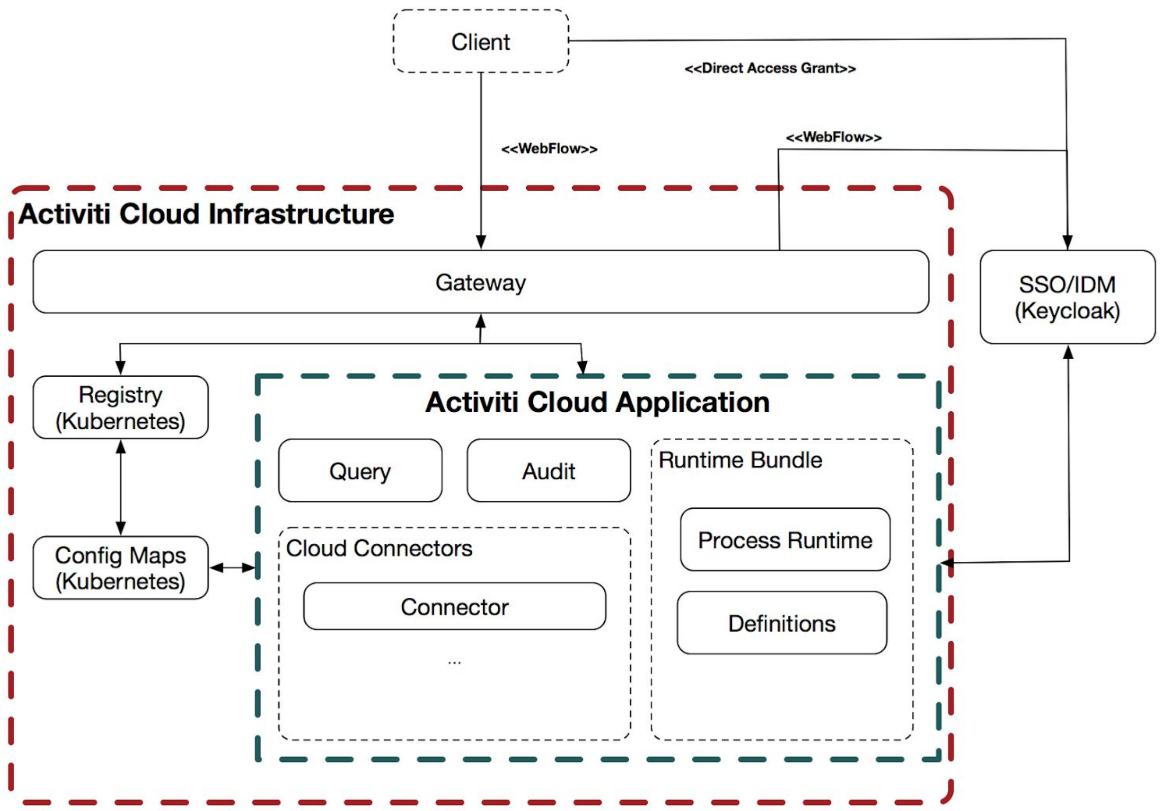
- Blob
- Queue
- Table

Alfresco Solution

Diagram


Alfresco Digital Business Platform
Application Development Framework
Integrations / Extensions
Open APIs and Open Standards
Alfresco Process Services
Alfresco Content Services
Alfresco Governance Services
Intelligence and Analytics
On-Prem, Cloud, Hybrid, Managed


App Designer
Admin App
Process Services UI
Microsoft Office
Alfresco Process Services
REST API
SharePoint Protocol
Process Engine
Rule Engine
Process Services logic
Identity Mgmt
LDAP / AD
Relational DB
MySQL/Oracle/Posgres/MS SQL Server/DB2
Analytics & Search
Elastic Search
File System
OS / S3



Install instruction

https://community.alfresco.com/community/bpm/blog/2018/08/13/getting-started-with-activiti-cloud-beta1

Architecture based Organization Structure and defined scope

- Kangaroo Infrastructure
  - Setup Microservice based Azure Cloud deployment
- Kangaroo Processes
  - Processes
  - Connectors
    - for internal processes
    - integrations with external systems
- Star NG NG
  - UI with basic functionalities and widget developers
  - Onboard Request types and Form creation and bundle it with process

Define boundaries