

**ABB DevOps Proposal**

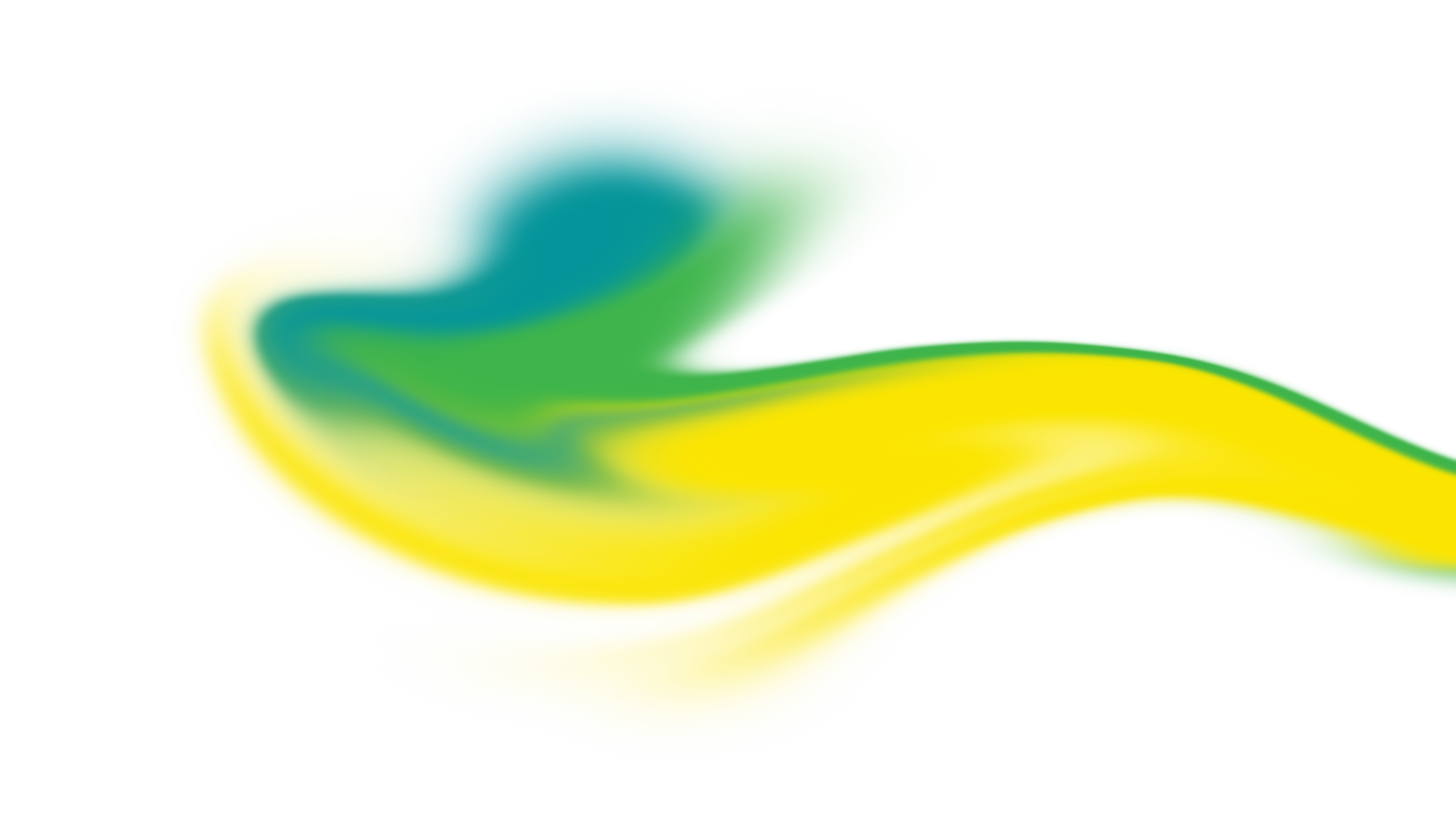


Table of Contents

[1 Our Understanding of Requirements and Scope 3](#_Toc115457836)

[1.1 Reference 3](#_Toc115457837)

[1.2 In scope 3](#_Toc115457838)

[1.3 Out of scope 3](#_Toc115457839)

[2 Executive Summary 3](#_Toc115457840)

[2.1.2 Technology Stack 6](#_Toc115457841)

[2.2 Proposed Architecture and Approach 7](#_Toc115457842)

[2.2.1 AKS Ecosystem 7](#_Toc115457843)

[2.2.2 Continuous Integration and Continuous Delivery 9](#_Toc115457844)

[2.2.3 Containerization 10](#_Toc115457845)

[2.2.4 Governance 10](#_Toc115457846)

[2.3 Assumptions 10](#_Toc115457847)

[2.4 General Assumptions 11](#_Toc115457848)

[2.5 Technology Assumptions 11](#_Toc115457849)

[3 Warranty 11](#_Toc115457850)

[4 Post-Delivery Support 11](#_Toc115457851)

[5 About Happiest Minds 11](#_Toc115457852)

# ****Our Understanding of Requirements and Scope****

ABB has a requirement to standardize and enable DevOps practices for their Edge Platform Business Applications.

At a high level the objective is to:

* Platform as code – AKS and related Azure Cloud components
* Enable version control system and define a branching strategy
* Continuous Integration and Delivery

## Reference

1. Based on the discussions via emails & calls between Happiest Minds and ABB teams

## In scope

Scope of work based on the calls with ABB team:

* Generic DevOps framework
  + The framework to have the capability to tie in CI CD and container ecosystem components (viz: Kubernetes, Azure DevOps pipelines, git, Prometheus, Grafana, SonarQube, infrastructure as code tool that spins up the container ecosystem etc.) and visualize and operate it at one plane (generic framework UI).
  + Generic DevOps framework will be setup and deployed on same tenant directory as that of the directory where AKS clusters need to be created (or are already running). One pilot project will be onboarded as par of this effort.
* Infrastructure as code (AKS ecosystem – to setup AKS and the container registry)
* Continuous integration and Continuous delivery (Onboard one example pilot project)
* Version control system, branching strategy, release promotion model

## Out of scope

* Integrating/set up/manage components/services/devices outside of Azure Cloud
  + IOT hub
  + edge devices
  + The application Microservice architecture or development
  + Disaster Recovery and backup management is not considered at present
  + Onboarding new projects apart from the pilot project shall not be considered as part of this proposal and can be address in subsequent SoW.

If needed, Happiest Minds can undertake the above activities, at additional cost.

# **Executive Summary**

## Generic DevOps Framework

ABB’s vision is to build a DevOps enabled platform for their Edge Platform Business Applications, ABB would like to implement a common framework that could be adopted by different project teams in order to standardize the tools and processes. Happiest Minds is pleased to propose a generic DevOps framework that would be configured by DevOps teams and used by developers. DevOps teams specify what resources start up with what environment or at what request. It would also enable to set base-line templates for application configurations and govern permissions. This helps developers to automate recurring tasks such as spinning up environments and resources and makes their setup easier to maintain by enforcing standards. Developer / project DevOps teams gain autonomy by changing configurations, deploying, spinning up fully provisioned environments, and rollback.

The Generic DevOps framework helps to achieve centralizing services and standardizing the tooling, it streamlines the development environments end to end. Reduce infrastructure complexity and provide standardization.

All the components starting with the version controls systems/git repositories to Kubernetes clusters can be visualised and managed from one single dashboard. The proposed generic framework also enables triggering builds from the Dashboard itself (CI CD)

The Generic framework would be built based on 5 core components

|  |  |
| --- | --- |
| **Core Component** | **Short Description** |
| Application Configuration Management | Manage application configuration in a scalable and reliable way |
| [Infrastructure Orchestration](https://internaldeveloperplatform.org/core-components/infrastructure-orchestration/) | Integrate with existing and future infrastructure |
| [Environment Management](https://internaldeveloperplatform.org/core-components/environment-management/) | Enable developers to create new environments whenever needed |
| [Deployment Management](https://internaldeveloperplatform.org/core-components/deployment-management/) | Implement a Continuous Delivery or even Continuous Deployment (CD) approach |
| [Role-Based Access Control](https://internaldeveloperplatform.org/core-components/role-based-access-control/) | Manage who can do what in a scalable way |

Application Configuration Management – The proposed framework would overcome the problem of Application Configuration Management by enabling ABB to manage environments and application configurations (viz: ingress resources for a given environment, resource quota and application specific Kubernetes configmaps) from a git repository – any commits to this repository would trigger an automated pipeline that would apply changes to the respective Kubernetes namespace. This has a significant impact on maintaining, debugging, and governing application configuration

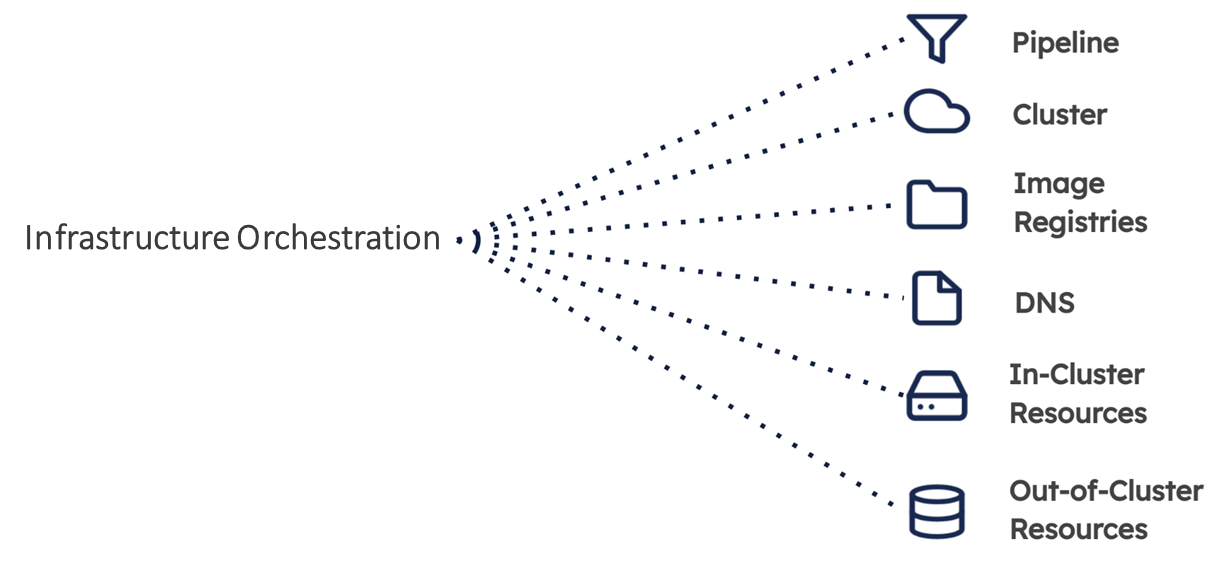
Challenges addressed:

* Configuration is often saved in script or YAML files. Maintaining these files can be hard even if applied an approach such as GitOps to them.
* Versioning configuration can be hard, especially since configuration very often needs to be altered depending on the environment to be used with. How to distinguish between environment-specific and non-environment-specific elements in the application configuration?
* Most current setups to manage application configuration do not allow for self-service from a developer. Thus, setting up a new environment for a feature branch or for a manual QA test needs involvement of an expert from the DevOps team.

Infrastructure Orchestration – The proposed framework would integrate with the existing infrastructure to enable Continuous Delivery or even Continuous Deployment (CD) processes. DevOps teams would be able to define which infrastructure is to be used whenever a new environment is created.

The framework follows a plug and play model, which allows to plug in and integrate various environments (AKS clusters and tenant namespaces for different environments). Other tools (viz: SonarQube, container registries, Infrastructure as code tools, Azure DevOps pipelines) which are part of the CI CD workflow, and the container ecosystem shall also plug into the framework and these islands of automation would be stitched together to enable seamless continuous application delivery to the target environment.

The framework has the capability to either import existing environments or create new environments.



Environment Management – The proposed framework would allow developers to self-serve new environments on demand. This removes a lot of bottlenecks and enables faster delivery. Each new environment is provisioned as defined by the DevOps team

Diagram

Description automatically generated

The proposed framework would tie together infrastructure, application configurations, and environments to boost the developer experience

* Self-service**:** The proposed generic DevOps framework would enable developers or teams to create new environments when needed. New environments would be created within the context of an application typically by cloning an existing deployment to a new environment. The developer can then modify the newly created environment as per the need (e.g., to test a new service or feature branch). All of this should happen based on infrastructure that is provided and maintained by the DevOps team.
* Environment types**:** The proposed framework would allow the DevOps team to define different types of environments. This ensures new environments are created with reasonable infrastructure requirements (e.g., small machines for development environments and a powerful setup for production or a load test environment)

Deployment Management – The proposed framework would enable teams to move to a Continuous Deployment (CD) process. It also provides a clear record of each deployment ever made which is great for audits and similar processes

While providing a streamlined developer experience is a key aspect of Deployment Management, there are other important aspects that needs to be supported like,

* Deployment (helm release) debugging support: What happens if the deployment as described in the last section fails? Debugging failed deployments can be a time-consuming task across multiple different systems with different logins and user interfaces/log files. The proposed framework would provide a consolidated view on the most important debugging information (e.g., deployment logs, container logs) for any current or even past deployment. This centralized information is a great starting point for debugging a problem and can save a lot of time and effort.
* Container image versioning: The proposed framework acts like a central memory for all deployments ever made. It typically stores all the information required to repeat a specific deployment. This not only significantly simplifies audit processes but also enables Git-like actions such as diffing two deployments or creating patches for deployments. It can also answer questions like: “Which version of any given service is running where?”, “In which environments was a certain version of a service deployed and tested before being released to production?”, etc. All deployed resources (Kubernetes resources or Kubernetes custom resources) will have a label on the metadata with this release version number.

Role based access control – The proposed framework would allow the DevOps team to manage access on a granular level. This can limit access to production to a small number of trusted people while allowing every engineer to create new development environments as needed.

Typical roles defined

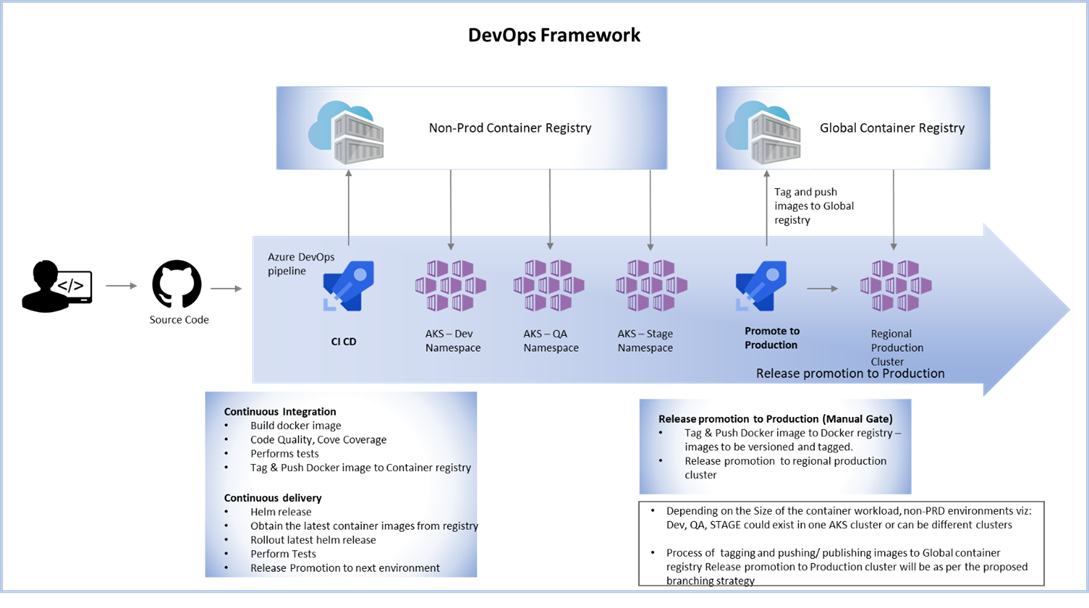
* **Member:** The role for a developer in a team. Members can typically access the applications they are working on.
* **Machine:** This role can have different names. It’s often a role that can be used for infrastructure integrations (e.g., for a CI pipeline pushing images into the Internal Developer Platform). It has very limited rights.
* **Manager:** The role for an engineering manager. Managers can typically invite users and manage the applications the team is responsible for. The scope might vary between different implementations.
* **Admin:** The role for the DevOps team or lead. Admins have full access to the entire functionality of the framework

## Technology Stack

Following are the primary technologies considered for this solution:

* Technology
  + Catalog service
  + Azure Pipelines, Terraform /Crossplane / Pulumi, Azure container registry, Kubernetes, Helm, Docker, Rancher,

## Proposed Architecture and Approach

The proposed generic framework would run on the same Azure account (same tenant directory) and can have SSO enabled.. One pilot project will be onboarded to the framework as part of this effort.

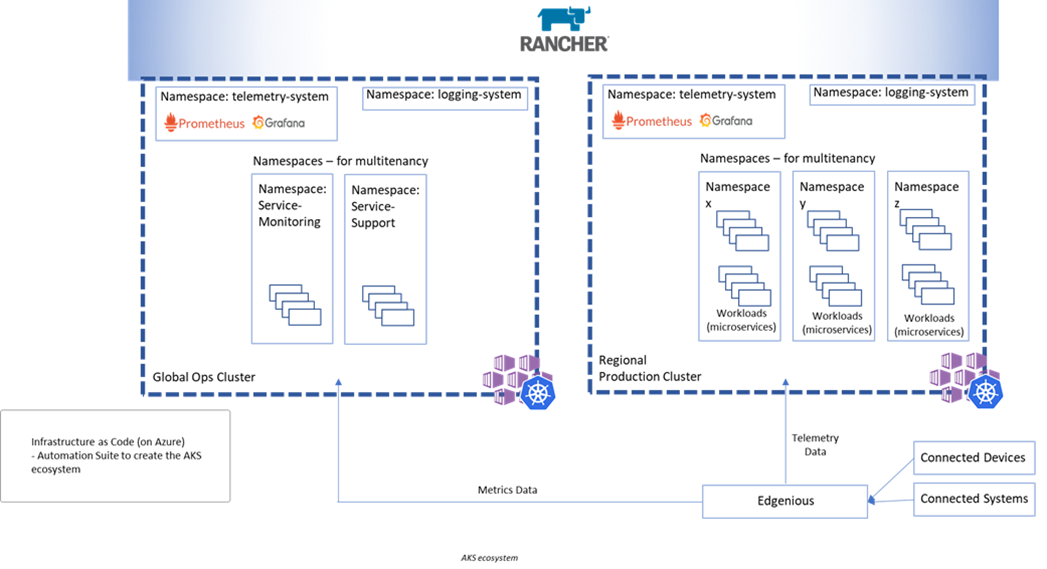
Infrastructure as Code (IaC) approach to be used to create the Azure Container ecosystem through the proposed framework. IaC modules would create of on-demand regional clusters and will be plugged into the generic framework and will be operated by users from the framework’s UI.

Azure repos would act as the version control system (git repos), CI CD will be enabled with the help of Azure pipelines. Any new commit to a repository would automatically trigger that CI CD pipeline which would build the container image, push it to the azure container registry (these docker images will be tagged with the git sha), run automated tests and deploy it to the dev cluster/namespace.

Release promotion to the subsequent environments(namespaces) following testing. Once the release branch is deployed to STG environment and all testing is complete and when it is ready for production release, a new “tag” needs to be created from the release branch (git repository), which would then publish the versioned container image tag to Global container registry and release ‘d be promoted to regional production AKS cluster.

Azure Repos, continuous integration and continuous delivery systems would also be plugged into the generic framework.

### AKS Ecosystem



AKS clusters will host the microservices. All the AKS clusters will be imported to Rancher and rancher would help to visualize administer test and develop on Kubernetes clusters. Clusters will also be plugged into the generic framework. RBAC rules will be in place to harden the ecosystem. Rancher will be used to manage the cluster workloads at one place – enables ease of cluster management.

Standard helm charts (the chart for the pilot project) that can be used as a reference point for creating helm charts for other projects across the organization and standardize the chart template across microservices/projects

Using the Generic framework, DevOps admin ‘d be able to create/configure all environments/AKS clusters or namespaces or container registries on demand without having to wait for an expert to create, configure and integrate the new environment to the CI/CD system. This ‘d also help reduce time and errors while different teams are involved in creating, managing, integrating these difference pieces/islands. Proposed framework to have the capability to create a fresh end to end DevOps practices enabled container platforms. i.e, the framework to possess the capability to either import existing environments or create new container environments.

#### Container Logging Solution

Any (new) AKS environment created using the generic framework, once that AKS cluster becomes available (to the framework, that is when the status of the AKS cluster becomes “available”) it would be configured to ship the “Pod” logs to an ELK cluster. i.e the generic framework would configure the beats on the provisioned AKS cluster to ship STOUT logs of all pods running on it to an Elasticsearch cluster.

#### Kubernetes Cluster Monitoring solution

When an AKS cluster is created using the generic framework, Prometheus and Grafana will be installed and configured on that cluster to enable Kubernetes cluster monitoring.

Grafana with Prometheus as the data source will act as the cluster monitoring solution for the AKS environment and for services running on AKS. Community built dashboards will be imported for monitoring & visualizing the microservices deployed in AKS and alert manager will be configured to handle alerts.

Monitoring solution will also be deployed in the AKS cluster in its own namespace, and it will be called monitoring-system.

### Continuous Integration and Continuous Delivery

Branching strategy, and developer workflow will be defined. Any commit to a branch will trigger a build and subsequent steps defined by the CI CD pipeline template. Release promotion will be as per the defined release strategy.

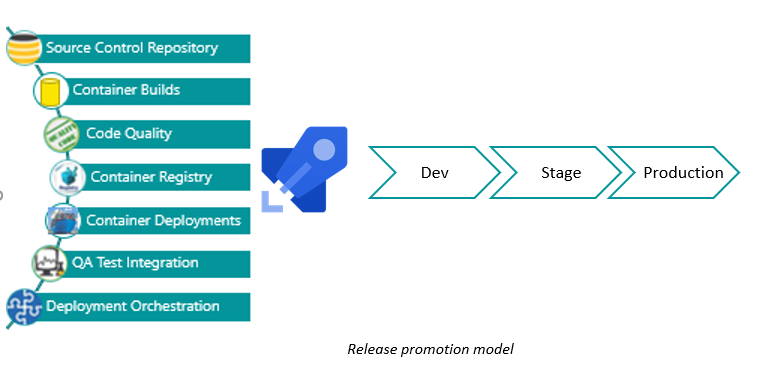
Azure DevOps pipelines will enable CI CD practices. code quality & code coverage an any automated test cases that may be involved will be integrated to the system.

Release promotion to production clusters – only the “tag” builds would deploy to production clusters. Container images will be built for each of the tag (create from a git branch) created (which would automatically trigger a CI CD pipeline) which would be identified separately from “branch” build and only these would be deploy to regional production clusters

Standard CI CD pipeline templates will enable standardization of the CI/CD process across projects in the platform. The Standard CI CD template would reference standard helm chart templates

Azure repos and pipelines will be plugged into and integrated with the proposed framework, enabling developers to create manage these pieces (repositories, CI CD pipelines, target environment) from the UI of the proposed framework

#### RELEASE PROMOTION MODEL AND BRANCHING STRATEGY



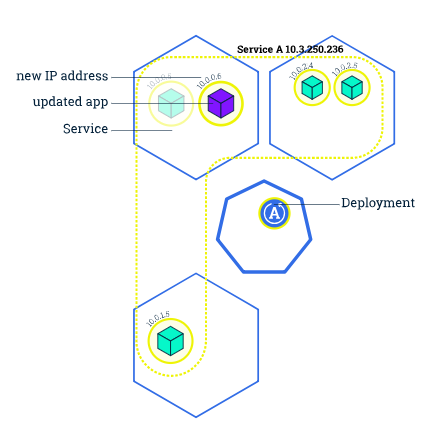
Azure repos would act as the version control system (git repos), CI CD will be enabled with the help of Azure pipelines. Any new commit to a repository would automatically trigger that CI CD pipeline which would build the container image, push it to the azure container registry (these docker images will be tagged with the git sha), run automated tests and deploy it to the dev cluster/namespace.

Release promotion to the subsequent environments(namespaces) following testing. Once the release branch is deployed to STG environment and all testing is complete and when it is ready for production release, a new “tag” needs to be created from the release branch(git repository), which would then publish the versioned container image tag to Global container registry and release ‘d be promoted to regional production AKS cluster

#### UPGRADE STRATEGY

Rolling update shall be followed during release upgrades and would be part of the helm release (defined by the helm chart)

Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones. The new Pods will be scheduled on Nodes with available resources.



### Containerization

Happiest Minds would onboard one container project to the generic framework. The Dockerfile and the helm chart of this example project ‘d be revied by the happiest minds team and then would share the review comments and create a merge request to update the Dockerfile and helm chart to fall in line with best practices and this project would act as a standard reference point for any new project that needs to be onboarded.

Some of the Key highlights of the Dockerfile shall be:

* Docker multi-stage builds
* Reduce the number of layers to what is needed.
* Choose the base image based on the need
* Container does not run as a privilege user

Helm Charts: A standard helm charts would help to eliminate unorganized workload in a multitenant Kubernetes cluster. Helm chart would pass required labels to all the resources created by it. Would also define the upgrade strategy and resource quota for the pods.

### Governance

The resources created in Azure would adhere to the following standard practices:

* Resource Tagging
* Resource Grouping
* Resource Naming Standards
* Identity and Role Based Access control
* Audit Policy

## Assumptions

### General Assumptions

1. Required Azure could access, and permissions shall be provided by ABB
2. Happiest Minds would develop the generic DevOps framework and onboard one pilot application to the platform. This pilot application shall be an existing app that ABB runs in their existing Kubernetes (AKS) cluster. CICD practices to be integrated when the app is onboarded to the generic framework.
3. The efforts considered for this proposal in restricted to the Generic DevOps framework and onboarding of one pilot project to the platform only. Happiest Minds can take up onboarding more projects in a different (and subsequent) proposal after the completion on the pilot project onboarding.
4. There will be a discovery phase prior to the actual work to understand the current AKS setup and then also to identify the pilot project which that shall be onboarded as part of this proposal effort.

### Technology Assumptions

1. Any DNS record creation, as part of Kubernetes (AKS) ingress resource shall be created by ABB – Happiest minds will provide the details of the ingress host and load balancer to ABB
2. Dockerfile(s) and helm cherts are available for these microservices. Happiest Minds shall onboard an existing containerized application (Pilot project) to the platform.
3. The container registries will be accessible to Azure pipelines and AKS clusters
4. Tools viz: Azure DevOps pipelines, git, Grafana, AKS clusters, SonarQube shall be integrated with the Generic framework

# ****Warranty****

One month of warranty will be provided after the completion of the project, for fixing all bugs reported that are related to software design.  Warranty does not cover design modifications due to requirement specification.

# ****Post-Delivery Support****

Any support required after the acceptance of the deliverables, can be taken up as part of post-delivery support. The quantum of support needed can be mutually discussed and agreed upon.  Post-delivery support will be charged on a monthly T&M basis.

# ****About Happiest Minds****

Happiest Minds is delighted to submit the proposal to ABB dated 28th Sep 2022. Happiest Minds is a next generation digital transformation, infrastructure & security, and product engineering Services Company with 5,000+ people, across 7 countries and serving over 195 active clients. Of these customers, 53 are Fortune-2000 / Forbes-200 / Billion $ corporations with 82% repeat business. We are a ‘Born Digital, Born Agile’ company with 96% of our business being digital in nature and 93% of it following agile practices. Happiest Minds enables digital transformation for enterprises and technology providers by delivering seamless customer experiences, business efficiency and actionable insights. We do this by leveraging a spectrum of disruptive technologies such as: artificial intelligence, blockchain, cloud, digital process automation, internet of things, robotics/drones, security, virtual/augmented reality, etc. Headquartered in Bengaluru, India, Happiest Minds has operations in the US, UK, Netherlands, Singapore, UAE, Canada, and Australia. Happiest Minds, under our promoter Ashok Soota, went IPO (Initial Public Offering) in September 2020. It was heavily oversubscribed with 315 % listing gains, reflecting our growth, profitability, corporate governance, management team and the value we bring with our mission of ‘Happiest People. Happiest Customers.’ Accolades: 1. Ranked #21 – India’s Best Companies to work for 2021 2. Ranked #63 – Asia’s Best Workplaces for 2021 3. India’s Top 50 Best Workplaces for Women for 2021 4. Golden Peacock Award for Business Excellence 2021 5. Glassdoor - 4.4 rating 6. Industry Analyst Acknowledgement for digital transformation led by IoT (Internet of Things), Analytics and Security (Gartner, Frost & Sullivan, Forrester, ISG, HFS)

Happiest Minds Technologies Limited (NSE: HAPPSTMNDS), a Mindful IT Company, enables digital transformation for enterprises and technology providers by delivering seamless customer experiences, business efficiency and actionable insights. We do this by leveraging a spectrum of disruptive technologies such as: artificial intelligence, blockchain, cloud, digital process automation, internet of things, robotics/drones, security, virtual/augmented reality, etc. Positioned as ‘Born Digital. Born Agile’, our capabilities span digital solutions, infrastructure, product engineering and security. We deliver these services across industry sectors such as automotive, BFSI, consumer packaged goods, e-commerce, edutech, engineering R&D, hi-tech, manufacturing, retail and travel/transportation/hospitality.

A Great Place to Work-Certified™ company, Happiest Minds is headquartered in Bangalore, India with operations in the U.S., UK, Canada, Australia and Middle East.