# **Argo CD for Canary Deployment on EKS**

| Version | Description | Revision date | Amendment/  Modification/  Deletion | Reviewer Name | Approver Name |
| --- | --- | --- | --- | --- | --- |
| 1.0 | Initial Version | 20th December 2023 | Amendment | Mohan Babu | Vyshnav |

## **Introduction:**

This runbook leads you through automated procedures, guaranteeing smooth shifts between application versions and reducing downtime to a minimum. Enhance your deployment strategy in Kubernetes environments by harnessing the strength and straightforwardness of Argo Rollouts.

## **Prerequisites:**

* Amazon EKS Cluster: Ensure you have an EKS cluster set up.
* kubectl: Install kubectl to interact with your EKS cluster.
* Helm: Install Helm, as Argo CD is often deployed using Helm charts.

## **Argo Rollouts:**

Argo Rollouts is a Kubernetes-native controller that enhances deployment strategies for applications. Offering advanced features like Blue-Green, Canary, and AB testing, Argo Rollouts simplifies and automates the process of releasing software updates. With traffic shifting, rollback capabilities, and integration with monitoring tools, Argo Rollouts empowers efficient and controlled deployments in Kubernetes environments, ensuring reliability and flexibility in managing application versions.

### **Manual Installation of Argo Rollouts Plugin:**

1. Download the latest release of the Argo Rollouts plugin:

| curl -LO https://github.com/argoproj/argo-rollouts/releases/latest/download/kubectl-argo-rollouts-linux-amd64 |
| --- |

Make sure to replace `Linux` and `amd64` with your operating system and architecture if you're not using Linux AMD64.

2. Move the downloaded binary to a directory in your `PATH`:

| chmod +x kubectl-argo-rollouts-linux-amd64  sudo mv kubectl-argo-rollouts-linux-amd64 /usr/local/bin/kubectl-argo-rollouts |
| --- |

If you don't have sudo permissions, you can move the binary to a directory that is in your `PATH`.

3. Verify the installation:

| kubectl argo rollouts version |
| --- |

This should display the version information for the Argo Rollouts plugin.

Now, you should be able to use `kubectl argo rollouts` commands, including `kubectl argo rollouts promote`.

Run the following commands to install Argo Rollouts.

| kubectl create namespace argo-rollouts |
| --- |

| kubectl apply -n argo-rollouts -f https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml |
| --- |

Check if we installed it successfully.

| kubectl get pod -n argo-rollouts |
| --- |

We can also install argo-rollouts through Helm.

| helm repo add argo https://argoproj.github.io/argo-helm  helm install my-release argo/argo-rollouts --set dashboard.enabled=true |
| --- |

This command installs Argo Rollouts, deploying its controller and a metrics server for performance data collection. Additionally, it enables the Argo Rollouts dashboard, offering a user-friendly interface to visualize and manage deployment rollouts.

## **Argo Rollouts Deployment Strategy:**

Argo Rollouts provides a robust deployment strategy for Kubernetes environments. Leveraging techniques like Blue-Green, Canary, and AB testing, facilitates seamless application updates. With features such as traffic shifting and automated rollback, Argo Rollouts ensures controlled, reliable, and flexible deployment processes, enhancing the overall management of application versions in Kubernetes clusters.

The configuration of Argo Rollouts has a strategy property for us to choose the deployment strategy we want, with two values ​​of blue-green and canary. In our case, it’ll be canary.

| apiVersion: argoproj.io/v1alpha1  kind: Rollout  metadata:  name: rollouts-demo  spec:  replicas: 5  strategy:  canary:  steps:  - setWeight: 20  - pause: {}  - setWeight: 40  - pause: {duration: 10}  - setWeight: 60  - pause: {duration: 10}  - setWeight: 80  - pause: {duration: 10}  revisionHistoryLimit: 2  selector:  matchLabels:  app: rollouts-demo  template:  metadata:  labels:  app: rollouts-demo  spec:  containers:  - name: rollouts-demo  image: argoproj/rollouts-demo:yellow  ports:  - name: http  containerPort: 8080  protocol: TCP  resources:  requests:  memory: 32Mi  cpu: 5m |
| --- |

| kubectl apply -f canary-rollout.yaml  Kubectl get rollout |
| --- |

When we create a Rollout, the Argo Rollouts implicit create a ReplicaSet for a current revision.

| kubectl get rs  Kubectl get pod |
| --- |

We can now establish a service to facilitate end-user connections to our rollout pods.

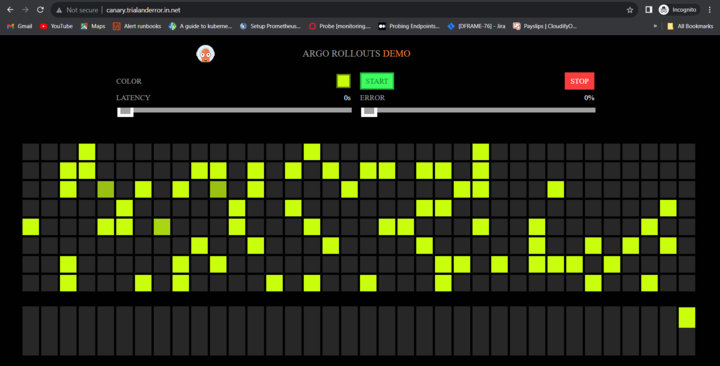
| apiVersion: v1  kind: Service  metadata:  name: rollouts-demo  spec:  type: LoadBalancer  ports:  - port: 80  targetPort: http  protocol: TCP  name: http  selector:  app: rollouts-demo --- |
| --- |

Ensure that Replica Set and Pod are running, next, we create a Service.

| kubectl apply -f service.yaml |
| --- |

After mapping the load balancer url with your dns records, Open the DNS name in the browser, we will see the UI below.

The following image illustrates Yellow version of the app in the DNS name in the browser:



Now, we change the image property of Rollout Object.

| apiVersion: argoproj.io/v1alpha1  kind: Rollout  metadata:  name: rollouts-demo  spec:  replicas: 5  strategy:  canary:  steps:  - setWeight: 20  - pause: {}  - setWeight: 40  - pause: {duration: 10}  - setWeight: 60  - pause: {duration: 10}  - setWeight: 80  - pause: {duration: 10}  revisionHistoryLimit: 2  selector:  matchLabels:  app: rollouts-demo  template:  metadata:  labels:  app: rollouts-demo  spec:  containers:  - name: rollouts-demo  image: argoproj/rollouts-demo:blue  ports:  - name: http  containerPort: 8080  protocol: TCP  resources:  requests:  memory: 32Mi  cpu: 5m |
| --- |

This Argo Rollout YAML defines a canary deployment for the rollouts-demo application with five replicas. The strategy involves incremental weight adjustments at 20%, 40%, 60%, and 80%, pausing for 10 seconds at each step ( 40%, 60%, and 80%). The canary deployment strategy involves an infinite pause at 20%, to promote to higher percentages, use the argo rollouts promote rollouts-demo command, which automatically progresses to the next weight defined in the canary strategy.

Updating Rollout Object. The revised version will have blue dots.

| kubectl apply -f canary-rollout.yaml |
| --- |

At this point, Argo Rollouts will create a new ReplicaSet for the new configuration.

| kubectl get rs |
| --- |

After we check the new ReplicaSet and see that all is well, next, we promote the new revision of ReplicaSet by updating the canary service to point to it, we run the following command.

| kubectl argo rollouts promote rollouts-demo |
| --- |

Now, Argo Rollouts update the canary service to point to the new ReplicaSet, after waiting (default 30 seconds), the old ReplicaSet is scaled down.

Argo Rollout provides us with a dashboard, which we can enable using kubectl or using quay.io/argoproj/kubectl-argo-rollouts container image.

| kubectl argo rollouts dashboard Argo Rollouts Dashboard is now available at localhost 3100 |
| --- |

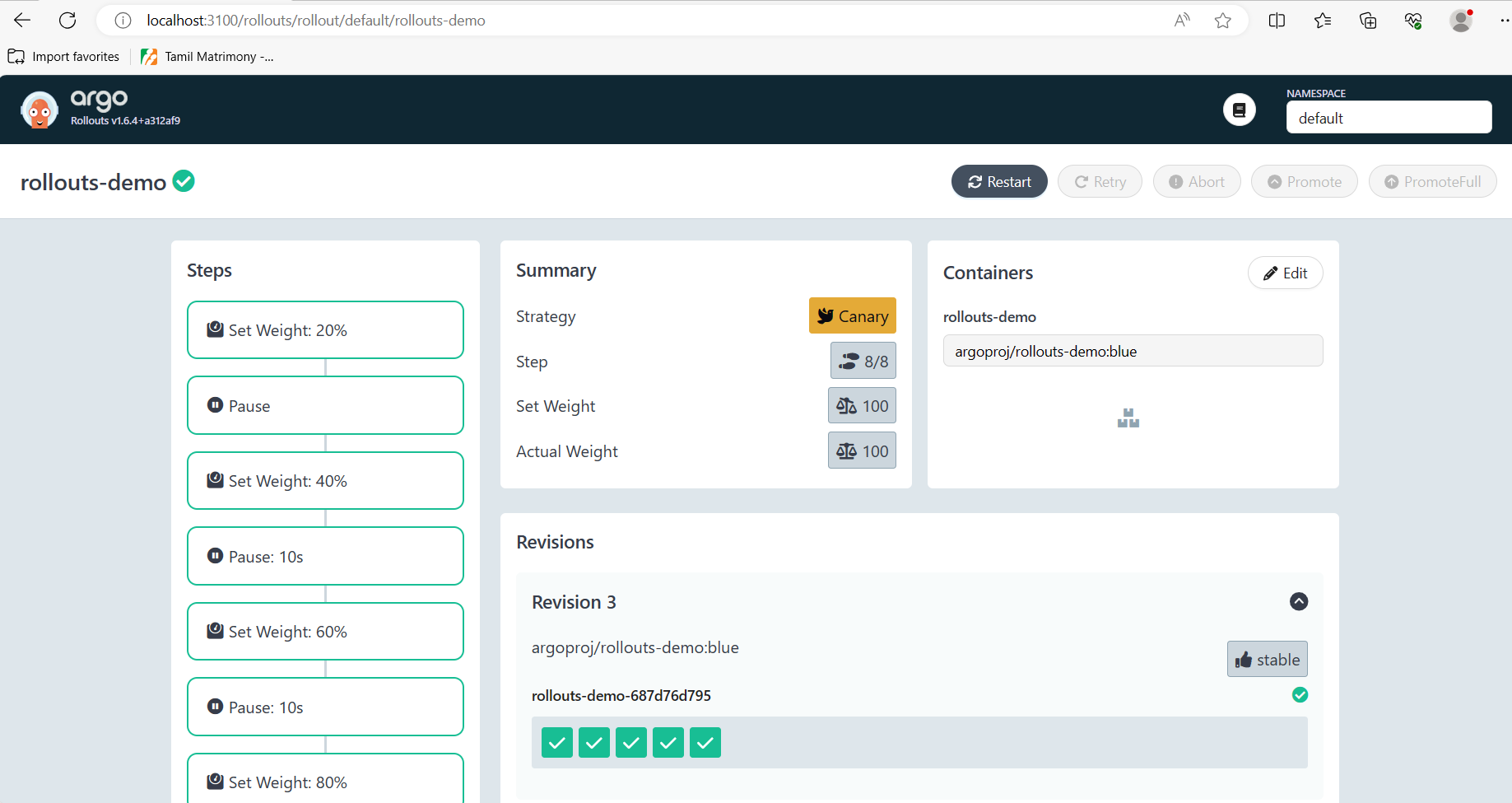
Go to localhost:3100 and we will see the dashboard of Argo Rollouts. Or one can use the following ingress setup for dashboard.

To set up an Ingress for dashboard, deploy the Ingress controller, map the DNS records in the domain name using the DNS name of the loadbalancer. Once it is configured, set up an ingress rule for the dashboard Service.

For example, the following can be configured as an ingress rule

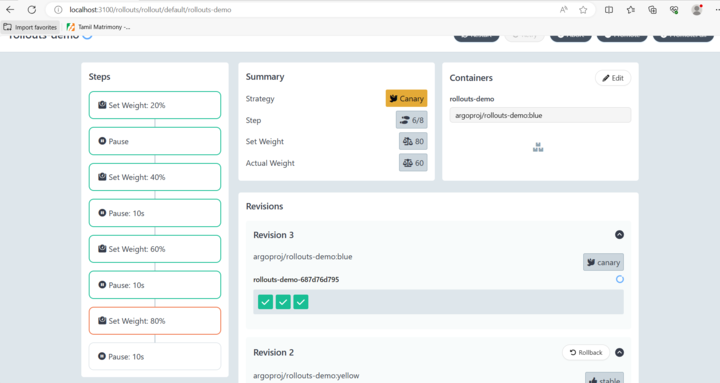
| apiVersion: networking.k8s.io/v1 kind: Ingress metadata:  name:argo-rollouts-dashboard  namespace: default  annotations:  nginx.ingress.kubernetes.io/rewrite-target: / spec:  ingressClassName: nginx  rules:  - host: < DNS name >  http:  paths:  - path: /dashboard  pathType: Prefix  backend:  service:  name:argo-rollouts-dashboard  port:  number: 8080 |
| --- |

---

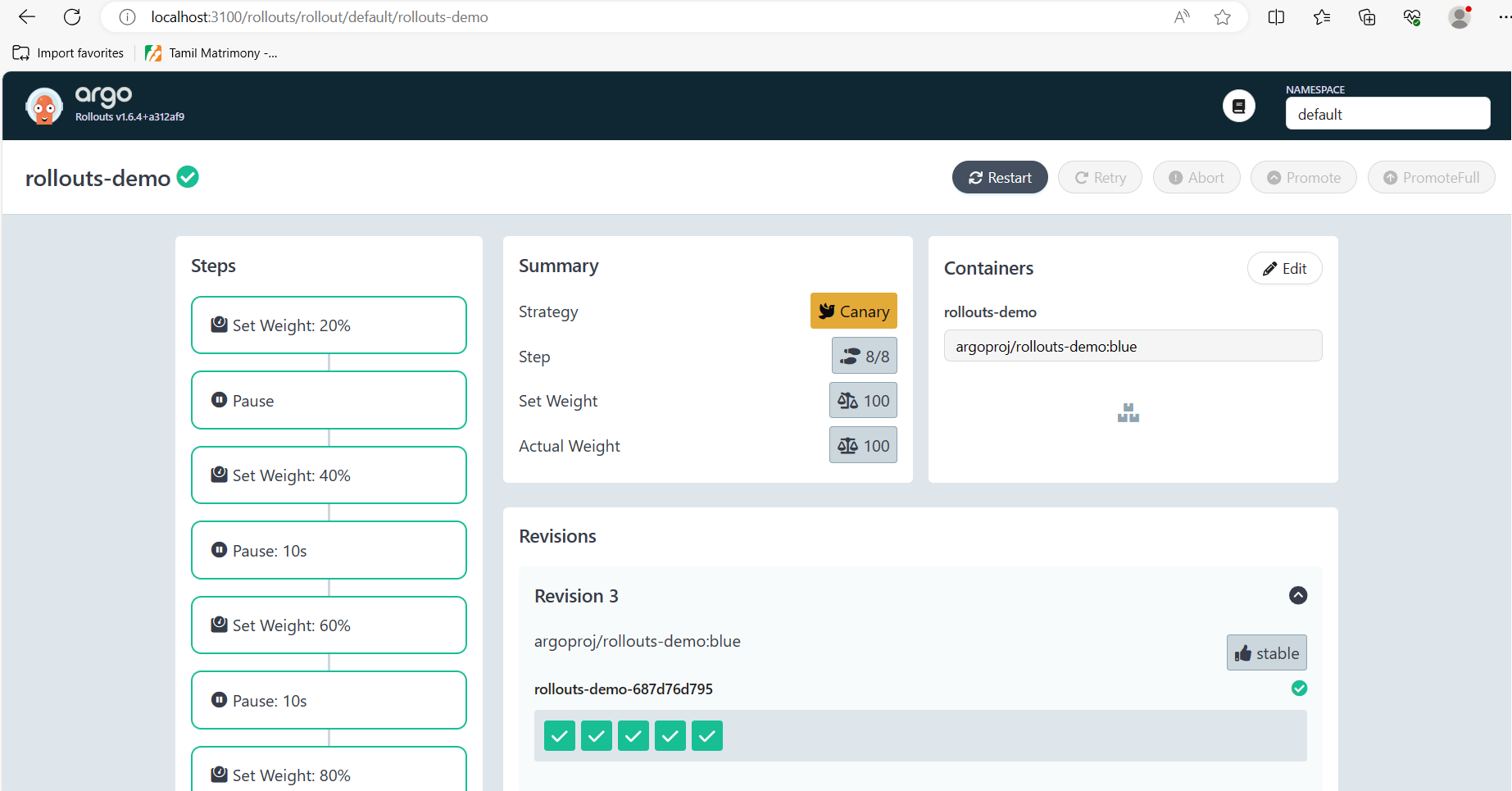


A new revision will be logged on the dashboard, pausing at specified weight intervals (20%, 40%, 60%, 80%), and subsequently, a complete replica set will be associated with the new revision.

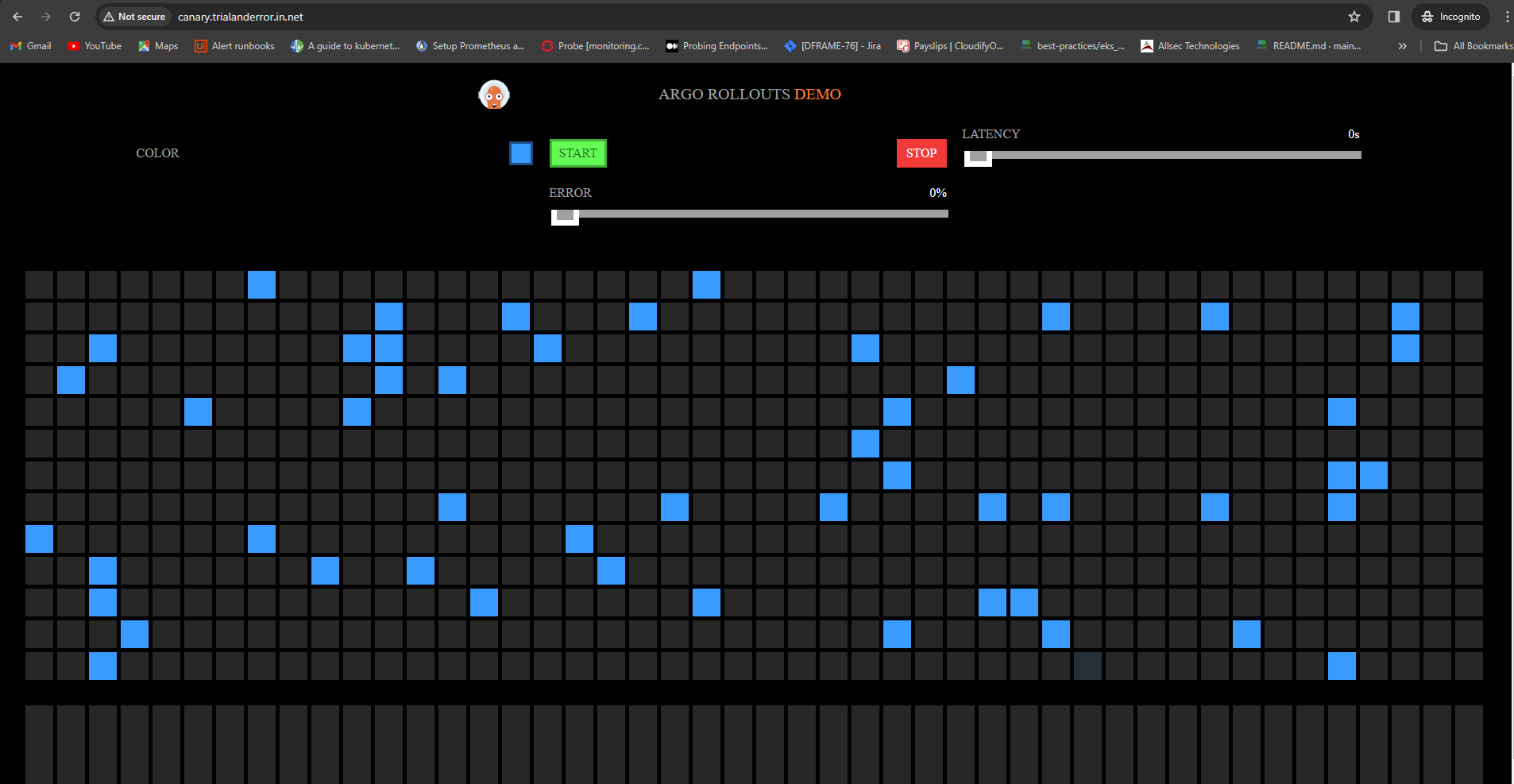
The following image illustrates 60% of the rollout in the argo rollouts dashboard:



The following image illustrates 100% of the rollout in the argo rollouts dashboard:



The following image illustrates blue version of the app in the DNS name in the browser:



## **Results:**

Argo Rollouts facilitates Canary deployments on Amazon EKS by offering a robust and declarative approach to Kubernetes application management. With Argo Rollouts, users can seamlessly transition between different versions of their applications, ensuring confidence and precision in the deployment process. The tool's user-friendly interface and adherence to GitOps principles simplify deployment, providing efficient control and visibility. Argo Rollouts' seamless integration with EKS ensures a scalable and reliable environment for managing Canary deployments, making it an excellent choice for Kubernetes-based continuous delivery workflows. In summary, Argo Rollouts enhances the Canary deployment experience on EKS, promoting a more automated and controlled application release process.

## **References:**

* <https://www.infracloud.io/blogs/progressive-delivery-argo-rollouts-canary-deployment/>
* <https://argo-rollouts.readthedocs.io/en/stable/>