This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

WebRTC Private Edition Guide

Deploy

8/12/2025

Contents

Learn how to deploy WebRTC Media Service (WebRTC) into a private edition environment.

Related documentation:

-
-
-

RSS:

- For private edition

Assumptions

- The instructions on this page assume you are deploying the service in a service-specific namespace, named in accordance with the requirements on Creating namespaces. If you are using a single namespace for all private edition services, replace the namespace element in the commands on this page with the name of your single namespace or project.
- Similarly, the configuration and environment setup instructions assume you need to create namespace-specific (in other words, service-specific) secrets. If you are using a single namespace for all private edition services, you might not need to create separate secrets for each service, depending on your credentials management requirements. However, if you do create service-specific secrets in a single namespace, be sure to avoid naming conflicts.

> **Important**
>
> Make sure to review Before you begin for the full list of prerequisites required to deploy WebRTC.

WebRTC uses blue-green model of deployment. It has the following main deployment principles:

- Both components - WebRTC Gateway and CoTurn Server - are deployed for each color and switched together
- Blue WebRTC Gateway is always configured to work with Blue CoTurn and green WebRTC Gateway is always configured to work with green CoTurn
- WebRTC have two FQDNs to reach active and inactive deployments:
  - **webrtc.domain.com** - active deployment. For example: webrtc.genesyshtcc.com
  - **webrtc-test.domain.com** - inactive deployment for tests. For example: webrtc-test.genesyshtcc.com

Deploy

You can deploy WebRTC using:

- Internal CoTurn Load Balancer or
- External CoTurn Load Balancer

Deploying WebRTC using internal CoTurn Load Balancer

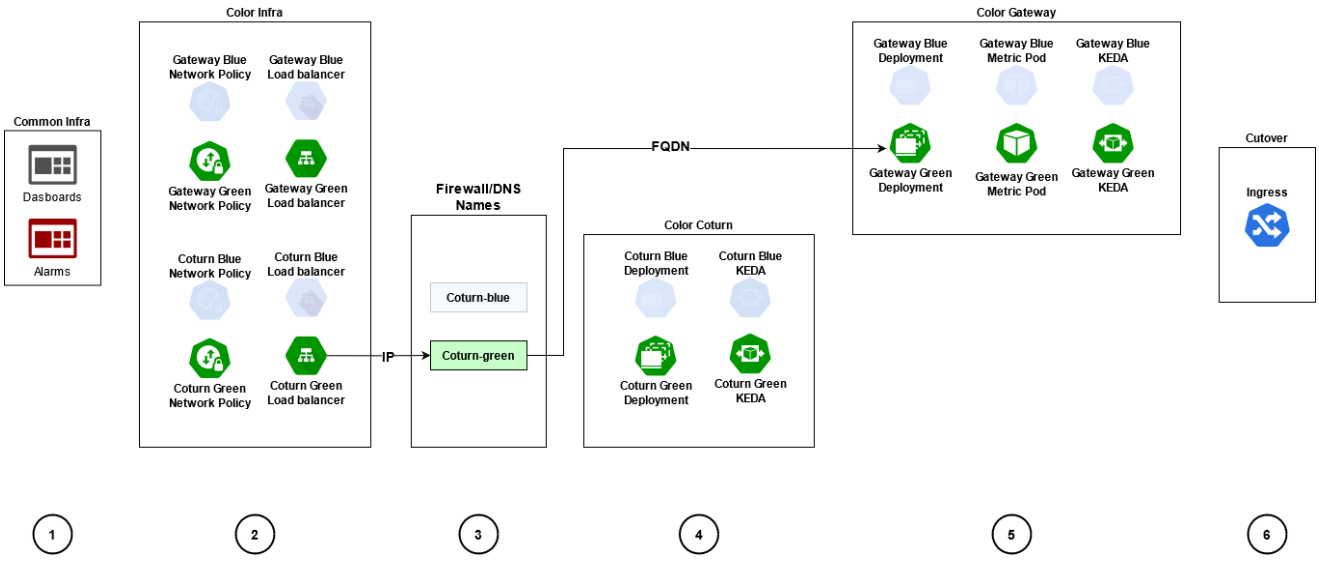
Initial deployment and Upgrade use the same sequence:

1. Deploy/upgrade inactive color of deployment
2. Make the cutover

You need to deploy the Color Infra package with CoTurn Load Balancer to get the IP address assigned automatically for the CoTurn Load Balancer by the infrastructure. Then, the infrastructure team should assign the IP to the CoTurn Load Balancer, create the FQDN for the IP and ensure that the IP is set in the firewall and is available from outside the cluster.

> **Important**
>
> The IP address assigned to the CoTurn Load Balancer must be external and available outside the cluster. Else, the media will not get through the WebRTC.

The following image shows the steps involved in deploying WebRTC using the internal CoTurn Load Balancer:



Follow the below steps to deploy WebRTC using internal CoTurn Load Balancer:

1. **Create common infrastructure elements such as dashboards and alarms**: This step deploys dashboards, alarms, and other common infrastructure elements.

   > **Important**
   >
   > You should perform this step even if you do not require the dashboard and alarms.

   Run the following command to create the common infrastructure elements:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=infra --
    set-string deployment.color="" webrtc-infra {HelmRepoPath}/webrtc-service --
    version={WebRTC Charts Version}
   ```

   Example:

   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=infra --set-
    string deployment.color="" webrtc-infra wrtchelmrepodevwestus2/webrtc-service --
    version=0.1.93 -n webrtc
   ```

2. **Create infrastructure elements for the deployment color**: This step deploys the infrastructure objects such as Turn Load Balancer, Gateway Service Object, Gateway Network Policies, and Turn Network Policies for the given color of deployment.
   You should also specify the INACTIVE color of deployment in this step.

   > **Important**
   >
   > You should configure the `deployment.coturnDeployment` option with the value `internal` in your values.yaml file.

   Run the following command to deploy the infrastructure objects:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=infra --
    set-string deployment.color={INACTIVE_COLOR} webrtc-infra-{INACTIVE_COLOR}
    {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=infra --set-
    string deployment.color=blue webrtc-infra-blue wrtchelmrepodevwestus2/webrtc-service
    --version=0.1.93 -n webrtc
   ```

3. **Get the IPs from the CoTurn Load Balancers, create DNS records and firewall rules**: This step gets the IP address from the Colurn Load Balancer created in Step 2. The name of LoadBalancer will be similar to: webrtc-coturn-service-{COLOR}.
   Create appropriate FQDN for this IP address in your DNS. This FQDN will be used by the WebRTC agents from outside the cluster to establish the RTP stream. Though you can use the IP address as it is, it is not the best practice to do so.

4. **Create CoTurn elements for the deployment color**: This step is to Upgrade/Deploy CoTurn for INACTIVE color.
   Run the following command to upgrade/deploy the INACTIVE color of deployment:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=coturn --
 set-string deployment.color={INACTIVE_COLOR} webrtc-coturn-{INACTIVE_COLOR}
 {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
```

Example:

```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=coturn --set-
 string deployment.color=blue webrtc-coturn-blue wrtchelmrepodevwestus2/webrtc-
 service --version=0.1.93 -n webrtc
```

5. **Create Gateway elements for deployment color using the information from Step 3**: This step is to Upgrade/Deploy Gateway for INACTIVE color. You shoud also specify the extenal FQDN of the CoTurn LoadBalancer in this step using the `gateway.turnExternalUriBlue` or `gateway.turnExternalUriGreen` options.
   Run the following command:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=gateway
    --set-string deployment.color={INACTIVE_COLOR} --set-string
    gateway.turnExternalUri{INACTIVE_COLOR}={COTURN FQDN INACTIVE_COLOR} webrtc-gateway-
    {INACTIVE_COLOR} {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example for Blue deployment:

   ```
   helm upgrade --install -f ./k8s/values.yaml--set-string deployment.type=gateway --set-
    string deployment.color=blue --set-string gateway.turnExternalUriBlue=turn-
    blue.ext.mydoamin.com  webrtc-gateway-blue wrtchelmrepodevwestus2/webrtc-service --
    version=0.1.93 -n webrtc
   ```

   Or, you can specify the IP of the Blue CoTurn Load Balancer

   ```
   helm upgrade --install -f ./k8s/values.yaml--set-string deployment.type=gateway --set-
    string deployment.color=blue --set-string gateway.turnExternalUriBlue=12.106.34.55
    webrtc-gateway-blue wrtchelmrepodevwestus2/webrtc-service --version=0.1.93 -n webrtc
   ```

6. **Create/update Ingress controller rules for Active/Inactive routing for Gateway deployments**: This step is to Install/upgrade ingress without changing the active color. The same step is used for the Cutover.

   > **Important**
   >
   > If you are deploying/upgrading `green`, specify the current ACTIVE color of deployment in the `deployment.color` option. Then specify `blue` and vice versa. If you deploying/upgrading `green` and specify `green` for the **cutover** step, the current active deployment will be switched to the just deployed/ upgraded `green`.

   You must perform this step even if you are not planning to make the cutover right now. This step is to upgrade the ingress and environment.

   Run the following command to create/upgrade Ingress controller rules:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=cutover
    --set-string deployment.color={ACTIVE_COLOR} webrtc-ingress {HelmRepoPath }/webrtc-
    service --version={WebRTC Charts Version}
   ```

   Example:

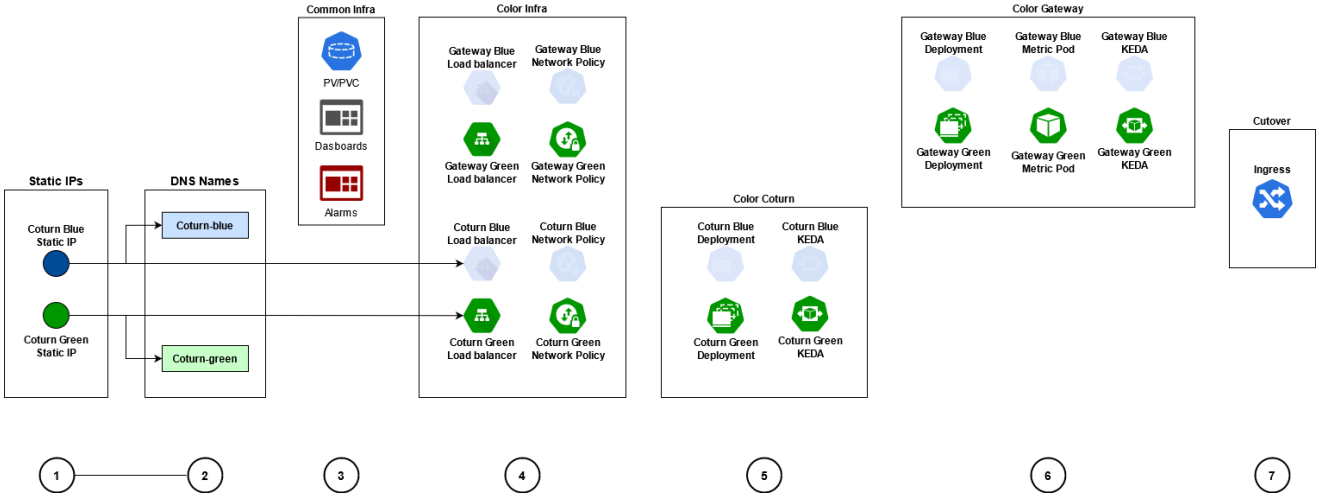
   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=cutover --
    set-string deployment.color=green webrtc-ingress wrtchelmrepodevwestus2/webrtc-
    service --version=0.1.93 -n webrtc
   ```

Deployment with external CoTurn Load Balancer

Initial deployment and Upgrade use the same sequence:

1. Deploy/upgrade inactive color of deployment
2. Make the cutover

The following image shows the steps involved in deploying WebRTC using the external CoTurn Load Balancer:



Follow the below steps to deploy WebRTC with external CoTurn Load Balancer

1. **Create static IPs for CoTurn**: This step is to specify the pre-created public IP for CoTurn Green in the `coturn.lbIpGreen` option and public IP for CoTurn Blue in the `coturn.lbIpBlue` option.
2. **Create DNS records for the created IPs**: This step is to specify the public FQDNs for CoTurn. Specify the pre-created public FQDN for CoTurn Green in the `gateway.turnExternalUriGreen` option and public FQDN for CoTurn Blue in the `gateway.turnExternalUriBlue` option.
3. **Create common infrastructure elements**: This step will deploy Persistent Volumes, Persistent Volume Claims, dashboards, alarms, and other common infrastructure elements.

> **Important**
>
> You need to run this step even if you are not using the dashboard and alarms.

Run the following command to create the infrastructure elements:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=infra --
 set-string deployment.color="" webrtc-infra {HelmRepoPath}/webrtc-service --
 version={WebRTC Charts Version}
```

Example:

```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=infra --set-
```

```
string deployment.color="" webrtc-infra wrtchelmrepodevwestus2/webrtc-service --
version=0.1.93 -n webrtc
```

4. **Create infrastructure elements for deployment color**: This step is to deploy the infrastructure objects such as Turn Load Balancer, Gateway Service Object, Gateway Network Policies, and Turn Network Policies for the given color of deployment.
   You must specify INACTIVE color of deployment for this step.

   > **Important**
   > Configure the `deployment.coturnDeployment` option with the value `external` in your `values.yaml` file.

   Run the following command to create the infrastructure elements:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=infra --
   set-string deployment.color={INACTIVE_COLOR} webrtc-infra-{INACTIVE_COLOR}
   {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=infra --set-
   string deployment.color=blue webrtc-infra-blue wrtchelmrepodevwestus2/webrtc-service
   --version=0.1.93 -n webrtc
   ```

5. **Create CoTurn elements for deployment color**: This step is to upgrade/deploy CoTurn for inactive color.
   Run the following command to specify the INACTIVE color of deployment:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=coturn --
   set-string deployment.color={INACTIVE_COLOR} webrtc-coturn-{INACTIVE_COLOR}
   {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=coturn --set-
   string deployment.color=blue webrtc-coturn-blue wrtchelmrepodevwestus2/webrtc-
   service --version=0.1.93 -n webrtc
   ```

6. **Create Gateway elements for deployment color**: This step is to upgrade/deploy the Gateway for inactive color.

   > **Important**
   > CoTurn DNS name is used for Gateway deployment as a parameter in the corresponding `values.yaml` file.

   Run the following command to specify the INACTIVE color of deployment:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=gateway
   --set-string deployment.color={INACTIVE_COLOR} webrtc-gateway-{INACTIVE_COLOR}
   {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

```
helm upgrade --install -f ./k8s/values.yaml--set-string deployment.type=gateway --set-
string deployment.color=blue webrtc-gateway-blue wrtchelmrepodevwestus2/webrtc-
service --version=0.1.93 -n webrtc
```

7. **Create/update Ingress controller rules for Active/Inactive routing for the Gateway deployments**: This step is to install/upgrade ingress without changing the active color. The sampe step is also used for the Cutover.

> **Important**
>
> If you are deploying/upgrading `green`, specify the current ACTIVE color of deployment in the `deployment.color` option which is `blue` and vice versa. If you deploying/upgrading `green` and specify `green` for the **cutover** step, the current active deployment will be switched to the just deployed/ upgraded `green`.

> **Important**
>
> You must perform this step even if you do not plan to make cutover right now. This step is to upgrade the ingress and environment.

Run the following command to create/upgrade Ingress controller rules:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=cutover
--set-string deployment.color={ACTIVE_COLOR} webrtc-ingress {HelmRepoPath }/webrtc-
service --version={WebRTC Charts Version}
```

Example:

```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=cutover --
set-string deployment.color=green webrtc-ingress wrtchelmrepodevwestus2/webrtc-
service --version=0.1.93 -n webrtc
```

Cutover

During cutover, it switches active color of deployment. This step should be performed only after you confirm that the newly installed/upgraded deployment is alive and functional. You must specify the current INACTIVE color of deployment in the `deployment.color` option - deployment that was just deployed/upgraded and tested. Run the following command to specify the cutover:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=cutover --set-
string deployment.color={INACTIVE_COLOR} webrtc-ingress {HelmRepoPath }/webrtc-service --
version={WebRTC Charts Version}
```

Example:

```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=cutover --set-string
deployment.color=blue webrtc-ingress wrtchelmrepodevwestus2/webrtc-service --version=0.1.93
-n webrtc
```

> **Important**
>
> You need to use PersistentVolume and PersistentVolumeClaim instead of HostPath logs of Gateway pods and CoTurn Pods.

Validate the deployment

Follow the given steps to validate the deployment.

1. **Verify PVCs are created and bound**

   ```
   kubectl get pvc
   ```

   Sample output:

   ```
   NAME                      STATUS   VOLUME                       CAPACITY   ACCESS
    MODES   STORAGECLASS     AGE
    webrtc-coturn-log-pvc     Bound    webrtc-coturn-log-volume     5Gi
    RWX            genesys-webrtc   110s
    webrtc-gateway-log-pvc    Bound    webrtc-gateway-log-volume    5Gi
    RWX            genesys-webrtc   110s
   ```

2. **Validate CoTurn and Gateway services**

   ```
   kubectl get svc
   ```

   Sample output:

   ```
   NAME                            TYPE           CLUSTER-IP      EXTERNAL-IP
    PORT(S)           AGE
    webrtc-coturn-service-blue     LoadBalancer   10.202.51.156   192.168.30.208
    443:31457/TCP     67m
    webrtc-gateway-service-blue    ClusterIP      10.202.47.170              80/TCP,8080/
    TCP   67m
   ```

3. **Query pods in the WebRTC namespace to confirm that pod is created, and in running status**

   ```
   kubectl get pods
   ```

   Sample output:

   ```
   NAME                                  READY   STATUS    RESTARTS   AGE
    webrtc-coturn-blue-b5db74c96-mh9jv    1/1     Running   0          4m20s
    webrtc-gateway-blue-d7ff45677-vbdg9   1/1     Running   0          86s
   ```

4. **Validate Ingress configuration**

   ```
   kubectl get ingress
   ```

   Sample output:

   ```
   NAME                 CLASS
    HOSTS
    ADDRESS   PORTS   AGE
    webrtc-ingress-int      webrtc.apps.vce-c0.eps.genesys.com,webrtc-test.apps.vce-
   ```

```
c0.eps.genesys.com             80      68s
```

5. Validate Ingress Edge route configuration

   ```
   kubectl get route
   ```

   Sample output:

   ```
   NAME                          HOST/PORT                                        PATH
    SERVICES                      PORT   TERMINATION   WILDCARD
   webrtc-gateway-service-blue   webrtc.apps.qrtph6qa.westus2.aroapp.io
   webrtc-gateway-service-blue   web    edge          None
   webrtc-ingress-int-cvdtt      webrtc.apps.qrtph6qa.westus2.aroapp.io        /
   webrtc-gateway-service-blue   web                  None
   webrtc-ingress-int-trcvh      webrtc.apps.qrtph6qa.westus2.aroapp.io        /blue
   webrtc-gateway-service-blue   web                  None
   webrtc-ingress-int-wf6x9      webrtc-test.apps.qrtph6qa.westus2.aroapp.io   /blue
   webrtc-gateway-service-blue   web                  None
   ```

6. **Query Ingress for** made available **WebRTC Web API**

   ```
   kubectl get ingress
   ```

   Copy the WebRTC API from the Ingress output:

   Sample output:

   ```
   NAME                 CLASS
    HOSTS
    ADDRESS   PORTS   AGE
    webrtc-ingress-int      webrtc.apps.vce-c0.eps.genesys.com,webrtc-test.apps.vce-
    c0.eps.genesys.com             80      3h26m
   ```

   Curl WebRTC "ping" API:

   ```
   curl -s webrtc.apps.vce-c0.eps.genesys.com/ping
   {"state":"up","version":"9.0.000.89","path":"blue"}
   ```