

オンプレとGCPのハイブリットでK3s/K8sで ゲームサーバーをオーケストレーション

@CNDW2025 学生スカラシップLT
by みずあめ

このスライドはここから読めます
<https://mizuame.works/slides/>

実際のデモクライアントの配布
やっています！！！！

<https://md.mizuame.app/s/jkiL6lEd9>

About Me

名前: 佐藤 良

年齢: 19歳

所属: 筑波大学情報学群情報科学類2年

ツイッター: @mizuameisgod

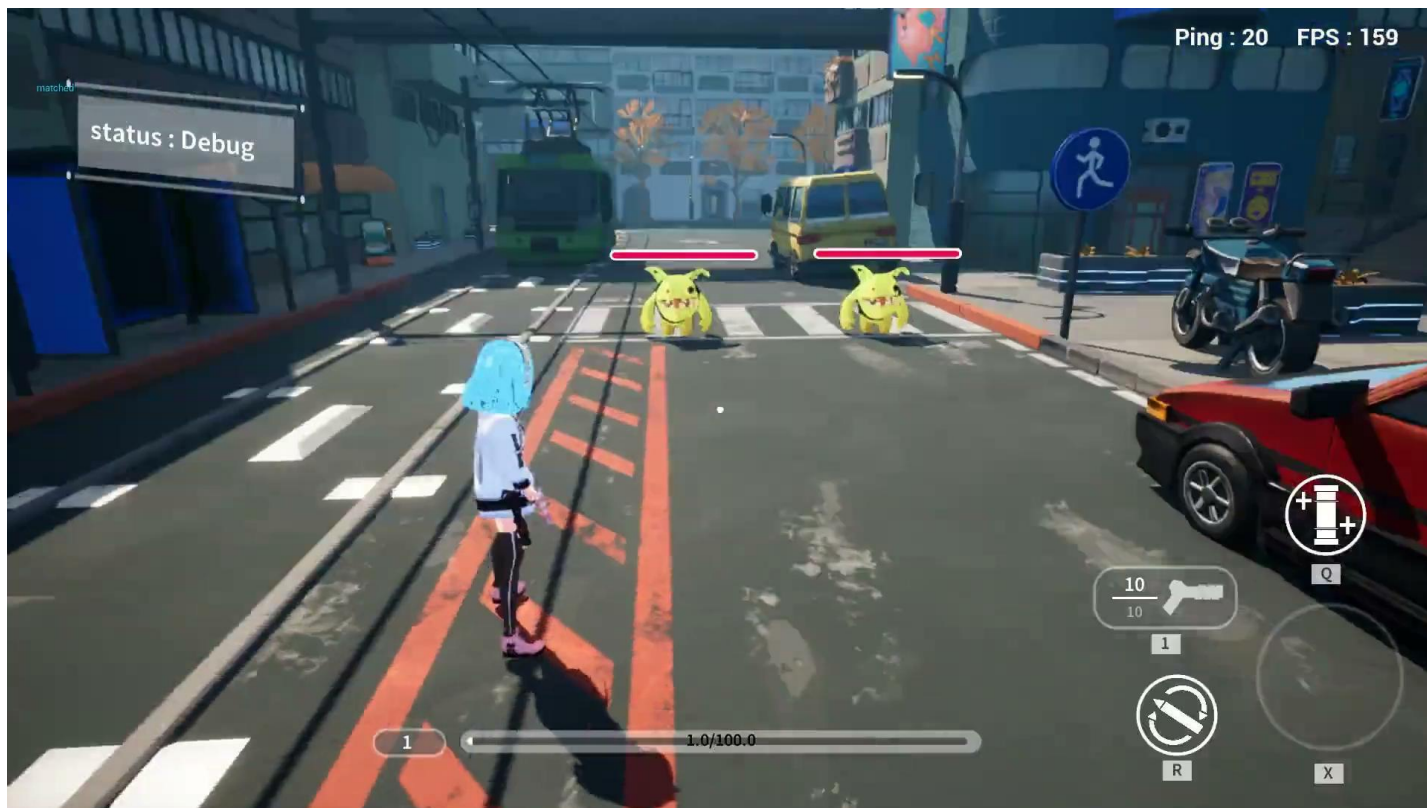
GitHub: @mizuamedesu

web: <https://mizuame.works/>



普段はゲームクライアントエンジニアとして、UnrealEngineやUnityをやっています

プレイ動画



クラウド、高い

- リアルタイム対戦ゲームサーバをサーバーレスでホストするのは難しい
- クラウド上でVMを沢山借りるとすぐ破産できる
- けど、オンプレでやると可用性に不安が(ex.停電)

→オンプレの計算資源を使いつつ、クリティカルな部分はクラウドのVMでやる
ハイブリッドK8sやりたい

問題が起きた時は、クラウドにVMを追加で生やし、フォールバックしたい。

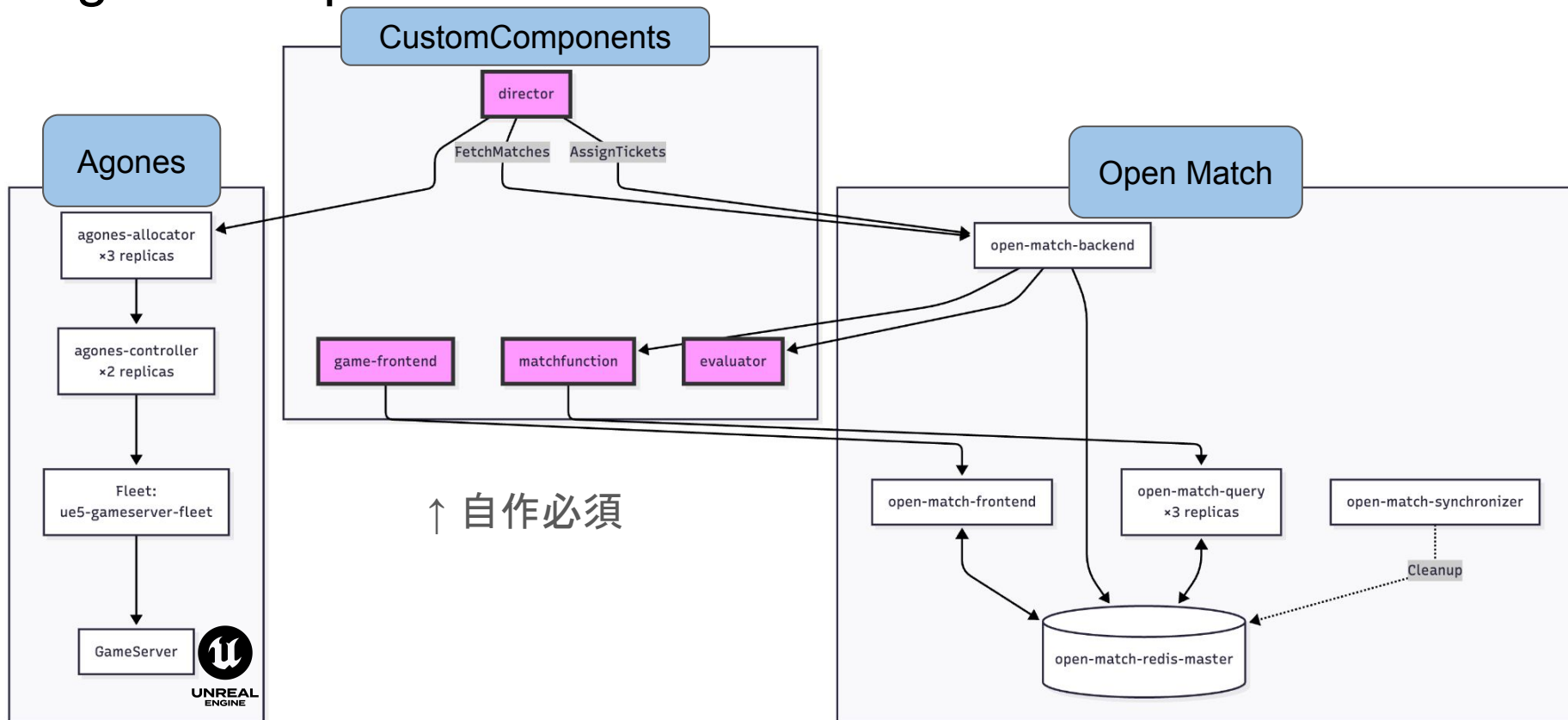
最終目標

- リレーサーバーを提供するSaaSはあるが、Dedicated serverを提供するSaaSはあまりない
- インディーゲーム界隈に、安価で提供できるDedicated server as a Serviceを作りたい
- 任意コード、バイナリの実行を許すため、セキュアな環境を作る必要がある
- 第一段階として、Kata Containersランタイムを使用したゲームランナーを構築する

AgonesとOpen Match

- Agones: 今回の場合はUnrealEngineのDedicated Serverを直接管理しているもの。サーバーのライフサイクルなどを管轄。
Googleなどが主にやってるOSS
- Open Match: 名前の通り、マッチメイキングを主に行う。クライアントとAgonesの橋渡しを実質的にやっている。(Open Match がマッチ結果を AgonesAllocator に渡し、サーバーを割り当てさせる)
Googleなどが主にやってるOSS

AgonesとOpen Match



AgonesとOpen Match

- director、match function、evaluator、game frontあたりは自分で作成する必要がある
- 簡易的なものを作成

mizuamedesu/**easy-open-match**



1

Contributor

0

Issues

0

Stars

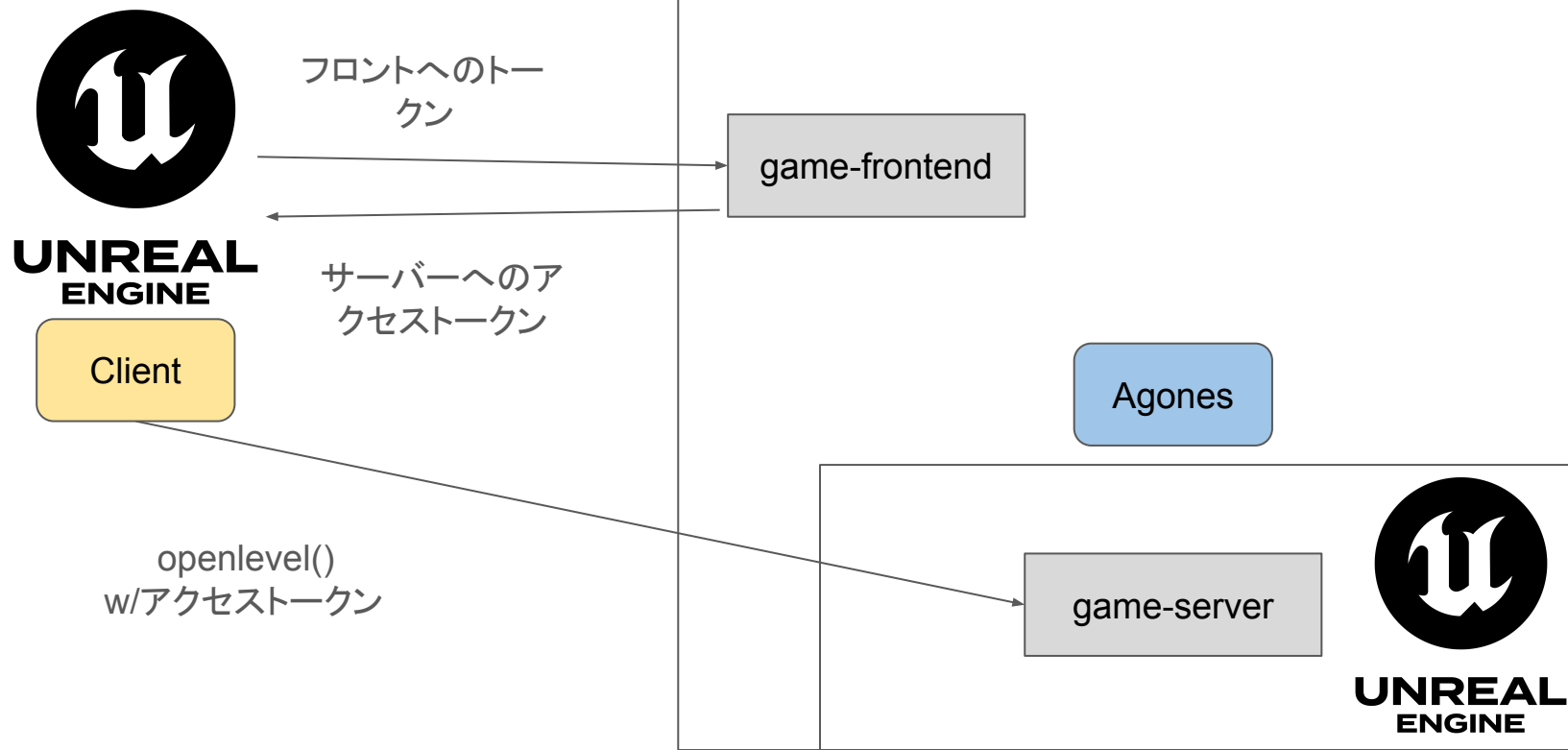
0

Forks



<https://github.com/mizuamedesu/easy-open-match>

AgonesとOpen Match



インチキクラスタはリアルタイムゲームサーバーと相性が良い？

Dedicated game servers also need a direct connection to a running game server process' hosting IP and port, rather than relying on load balancers. These fast-paced games are extremely sensitive to latency, which a load balancer only adds more of. Also, because all the players connected to a single game server share the in-memory game simulation state at the same time, it's just easier to connect them to the same machine.

専用ゲームサーバーの場合、ロードバランサーに依存するのではなく、稼働中のゲームサーバープロセスが動作しているホストのIPアドレスとポートに直接接続する必要があります。このようなリアルタイム性が求められるゲームは遅延に対して非常に敏感であり、ロードバランサーはむしろ遅延を増大させる要因となります。さらに、単一のゲームサーバーに接続しているすべてのプレイヤーが同時にメモリ内のゲーム状態を共有するため、同じマシンに直接接続する方がはるかに効率的です。

<https://cloud.google.com/blog/products/containers-kubernetes/introducing-agnes-open-source-multiplayer-dedicated-game-server-hosting-built-on-kubernetes>

インチキクラスタはリアルタイムゲームサーバーと相性が良い？

- 1マッチ=1Pod、NodePortでそのまま公開する
→ 最終的にクライアントはゲームサーバーにIP+Portで直で接続する
負荷分散はフロントのマッチセッションの所だけを考えれば良い
- 内部でAgonesがゲームサーバーのライフサイクルを管理できれば良いので、
インチキクラスタ構成と都合が良い

オンプレ/クラウドハイブリッドK8s



163.220.236.xxx

- FRR ルーター × 2(オンプレ)

i3-7100 4Core 4GB RAM /128GB HDDUbuntu24.04



163.220.236.xxx

- Worker(オンプレ)

xeon-2699v4 22Core 32GB RAM /512GB SSD Ubuntu24.04



10.10.0.xxx

- Master(GCP/control-plane)

e2-medium 2vCPU 4GB RAM /40GB Disk Ubuntu22.04

オンプレ/クラウドハイブリッドK8s

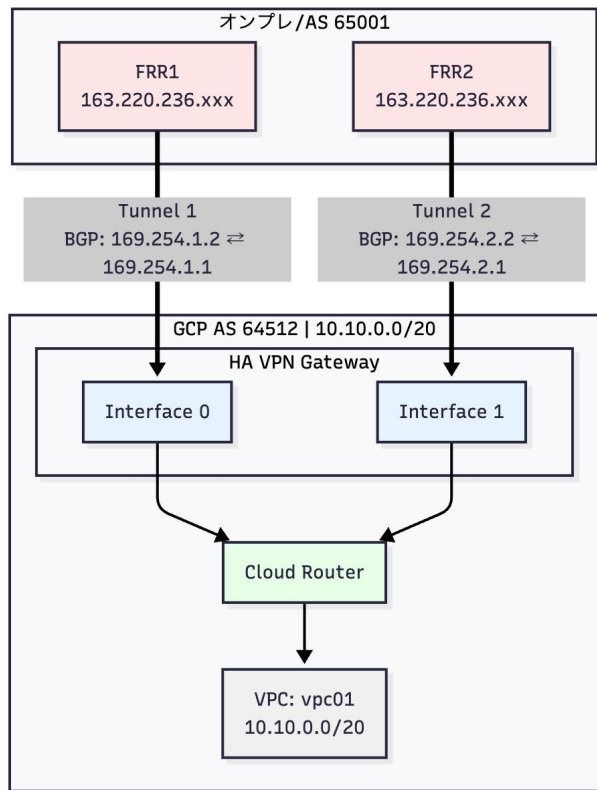


10.10.0.xxx

```
mizuame@k8s-gcp-test:~$ kata-runtime kata-check
No newer release available
ERROR[0000] CPU property not found arch=amd64 description="Virtualization support" name=vmx pid=9212 source=runtime type=flag
ERROR[0000] Module is not loaded and it can not be inserted. Please consider running with sudo or as root arch=amd64 module=kvm_intel name=kata-runtime pid=9212 source=runtime
ERROR[0000] kernel property kvm_intel not found arch=amd64 description="Intel KVM" name=kvm_intel pid=9212 source=runtime type=module
ERROR[0000] Module is not loaded and it can not be inserted. Please consider running with sudo or as root arch=amd64 module=kvm name=kata-runtime pid=9212 source=runtime
ERROR[0000] kernel property kvm not found arch=amd64 description="Kernel-based Virtual Machine" name=kvm pid=9212 source=runtime type=module
ERROR[0000] ERROR: System is not capable of running Kata Containers arch=amd64 name=kata-runtime pid=9212 source=runtime
ERROR: System is not capable of running Kata Containers
```

- Worker(GCP/オンプレのフォールバック先/一時ノード)
n2-standard-4 4vCPU 16GB RAM /50GB Disk Ubuntu22.04
- Kata Containersの使用にはkvmに対応している必要がある。いわゆる、Nested Virtualization
- GCPのN1/N2VMなどが対応

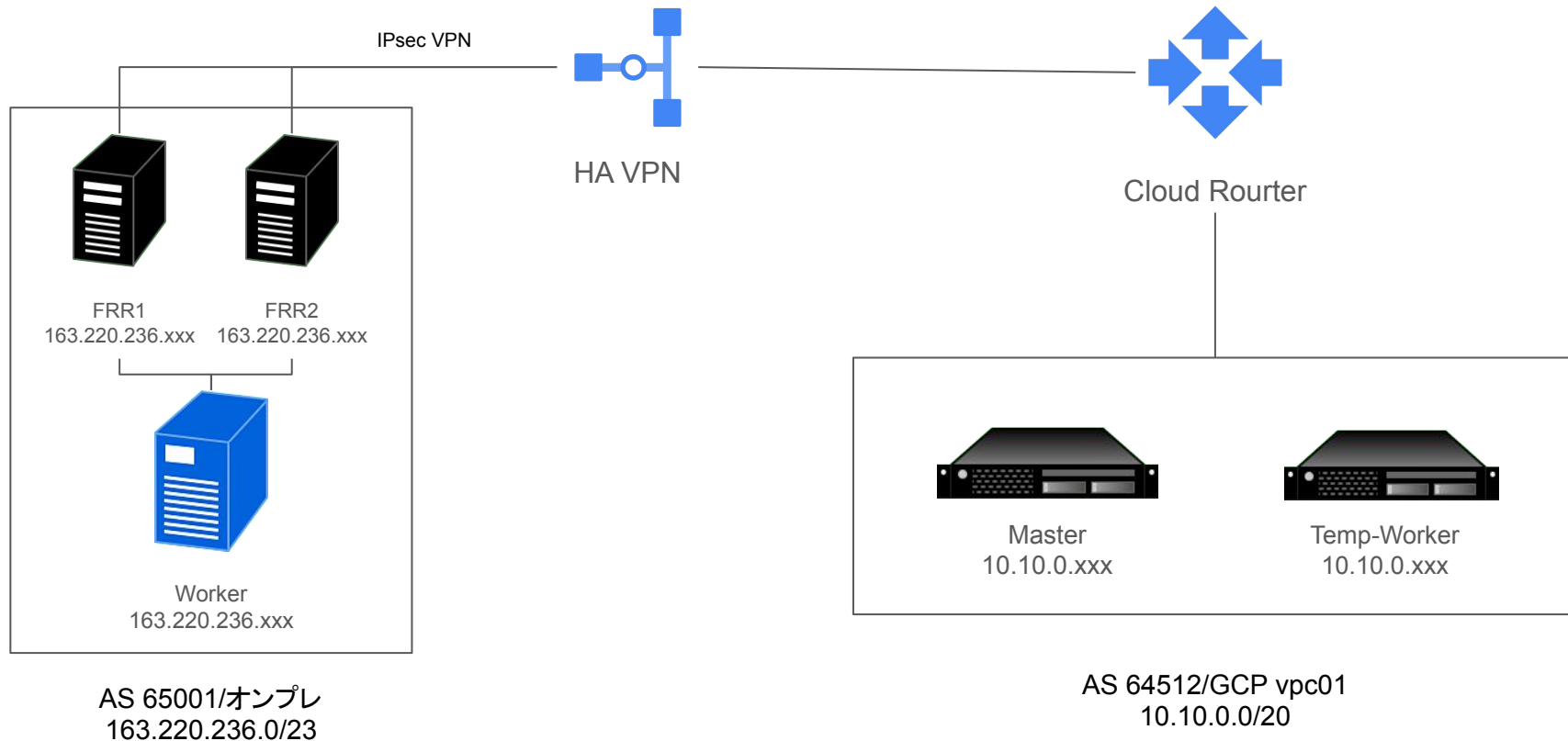
オンプレ/クラウドハイブリッドK8s



オンプレミス環境 (AS 65001) の FRR ルーター 2 台と GCP (AS 64512) の HA VPN Gateway の間で、冗長構成の IPsec VPN トンネルを確立

BGP ピアリングを行い、オンプレミス側は 163.220.236.0/23 を、GCP 側は 10.10.0.0/20 のサブネットを相互に広報

オンプレ/クラウドハイブリッドK8s



闇のノードオペレーター

- オンプレを監視し、疎通が取れなくなった場合GCPにVMを作成する、闇のオペレーターを作成
- 復旧時は自動で一時VMをクリーンアップ

mizuamedesu/**hybrid-
node-operator**



1

Contributor

0

Issues

1

Star

0

Forks



<https://github.com/mizuamedesu/hybrid-node-operator>

闇のノードオペレーター

- onpremiseラベルを監視対象のノードに付与: 今回はuc-k8s4p
- gcp-permanentを永続ノードに付与: gcp-master

→ここに監視用オペレーターが配置される

```
mizuame@k8s-master-1:~$ kubectl get nodes -o wide
NAME                                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
k8s-master-1.asia-northeast1-a.c.k8s-test-474912.internal  Ready    control-plane  14d   v1.31.13  10.10.0.23    <none>         Ubuntu 24.04.3 LTS   6.14.0-1018-gcp  containerd://1.7.28
uc-k8s4p                            Ready    <none>      14d   v1.31.13  163.220.236.54 <none>         Ubuntu 24.04.3 LTS   6.8.0-86-generic  containerd://1.7.28
```

```
mizuame@k8s-master-1:~$ kubectl get nodes -o wide
NAME                                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
gcp-temp-uc-k8s4p-1763028338        Ready    <none>      38s   v1.31.13  10.10.0.42    35.243.101.18 Ubuntu 22.04.5 LTS   6.8.0-1042-gcp      containerd://1.7.28
[k8s-master-1.asia-northeast1-a.c.k8s-test-474912.internal  Ready    control-plane  14d   v1.31.13  10.10.0.23    34.146.14.104 Ubuntu 24.04.3 LTS   6.14.0-1018-gcp      containerd://1.7.28
uc-k8s4p                            NotReady <none>      14d   v1.31.13  163.220.236.54 <none>         Ubuntu 24.04.3 LTS   6.8.0-86-generic    containerd://1.7.28
mizuame@k8s-master-1:~$ █
```


闇のノードオペレーター

[2025-11-13 10:05:06] Node event detected: node=uc-k8s4p,
ready=False

- オンプレノードが NotReady と判断され、フェイルオーバー処理が開始される

[2025-11-13 10:05:07] Deleting completed NodeFailover resource
[2025-11-13 10:05:07] Creating NodeFailover resource for NotReady
node uc-k8s4p
[2025-11-13 10:05:07] NodeFailover resource created

- 新しいフェイルオーバー処理がセットアップされる

闇のノードオペレーター

[2025-11-13 10:05:38] Creating GCP VM for failed node uc-k8s4p
[2025-11-13 10:05:38] Created bootstrap token
[2025-11-13 10:05:38] Successfully calculated CA certificate hash
[2025-11-13 10:05:59] Creating GCP instance gcp-temp-uc-k8s4p-1763828338
[2025-11-13 10:05:59] Successfully created instance
[2025-11-13 10:06:29] Node gcp-temp-uc-k8s4p-1763828338 has joined the cluster
[2025-11-13 10:06:29] Labeled node: node-type=gcp-temporary, node-location=gcp, hardware=game-runner

- オンプレノードが NotReady と判断され、フェイルオーバー処理が開始される。
- VMの作成が完了するとKubernetes クラスタへjoin、ラベル付与

闇のノードオペレーター

- 事前に全て(K8s、Kata Containersなど)がセットアップされたカスタムコンピュータイメージを持っている

```
gcloud compute images list --no-standard-images
```

NAME	PROJECT	FAMILY	DEPRECATED	STATUS
k8s-kata-node-v1		k8s-kata-node		READY

- このイメージを元に、VMを作成し、Joinさせる
 - kubeletに provider-id を設定することでcloud-controller-manager経由で ExternalIPを取得できる
- ```
--cloud-provider=external
--provider-id=gce://testserver-4343/asia-northeast1-b/gcp-temp-uc-k8s4p-x
xxxx
```



## 闇のノードオペレーター

- `create_bootstrap_token()`  
Kubernetes一時トークン作成
- `get_ca_cert_hash()`  
クラスタCA証明書のSHA256ハッシュ取得
- Startup Script生成  
`generate_startup_script(token, ca_hash)`  
メタデータ取得 (PROJECT, ZONE, INSTANCE\_NAME)  
kubelet設定 (cloud-provider=external, provider-id)  
`kubeadm join`



# 闇のノードオペレーター

```
mizuame@k8s-master-1:~$ kubectl get nodes -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME
k8s-master-1.asia-northeast1-a.c.k8s-test-474912.internal Ready control-plane 14d v1.31.13 10.10.0.23 <none> Ubuntu 24.04.3 LTS 6.14.0-1018-gcp containerd://1.7.28
uc-k8s4p Ready <none> 14d v1.31.13 163.220.236.54 <none> Ubuntu 24.04.3 LTS 6.8.0-86-generic containerd://1.7.28

mizuame@k8s-master-1:~$ kubectl get nodes -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME
gcp-temp-uc-k8s4p-1763028338 Ready <none> 38s v1.31.13 10.10.0.42 35.243.101.18 Ubuntu 22.04.5 LTS 6.8.0-1042-gcp containerd://1.7.28
k8s-master-1.asia-northeast1-a.c.k8s-test-474912.internal Ready control-plane 14d v1.31.13 10.10.0.23 34.146.14.104 Ubuntu 24.04.3 LTS 6.14.0-1018-gcp containerd://1.7.28
uc-k8s4p NotReady <none> 14d v1.31.13 163.220.236.54 <none> Ubuntu 24.04.3 LTS 6.8.0-86-generic containerd://1.7.28
```

- `create_instance(vm_name, startup_script)`  
カスタムイメージ使用: k8s-kata-node-v1  
metadata.startup-script に設定  
VM起動 → スクリプト自動実行
- Ready状態まで自動でなる
- EXTERNAL-IPが振られている  
→ Agonesが自動でグローバルIP:Portを振ってくれる



## 闇のノードオペレーター

```
mizuame@k8s-master-1:~$ kubectl get gameservers -n game -o wide
```

| NAME                             | STATE     | ADDRESS       | PORT | NODE                         | AGE |
|----------------------------------|-----------|---------------|------|------------------------------|-----|
| ue5-gameserver-fleet-z2jhh-5prsk | Scheduled | 34.146.94.130 | 7254 | gcp-temp-uc-k8s4p-1763029184 | 43s |
| ue5-gameserver-fleet-z2jhh-hdjcm | Scheduled | 34.146.94.130 | 7995 | gcp-temp-uc-k8s4p-1763029184 | 43s |

```
mizuame@k8s-master-1:~$
```

Onprem node uc-k8s4p still NotReady, applying out-of-service taint

Added taint to node uc-k8s4p

Set condition TaintApplied=True for uc-k8s4p

Applied out-of-service taint to uc-k8s4p

- VMがJoinした後、オンプレに  
node.kubernetes.io/out-of-service:NoExecute(taint付与)
- GCPの一時VMに各種サーバーが再配置される(画像参照)



## 闇のノードオペレーター

[2025-11-13 10:13:56] Node uc-k8s4p has re-joined the cluster  
[2025-11-13 10:13:56] Labeled node: node-type=onprem, node-location=onprem,  
hardware=game-runner  
2025-11-13 10:13:56] Node event detected: uc-k8s4p ready=True  
[2025-11-13 10:13:56] Onprem node recovery detected: uc-k8s4p  
[2025-11-13 10:13:56] NodeFailover phase changed: Active → Recovering

- `remove_node_taint()`

復旧を検知次第、オンプレノードの out-of-service taint削除  
GameServerがオンプレに戻れるようになる

- GCP一時ノードをcordon(新規Pod配置停止)



## 闇のノードオペレーター

```
[mizuame@k8s-master-1:~$ kubectl get gameserver -n game
NAME STATE ADDRESS PORT NODE AGE
ue5-gameserver-fleet-z2jhh-5vbqv Scheduled 163.220.236.54 7398 uc-k8s4p 24s
ue5-gameserver-fleet-z2jhh-hksj7 Ready 163.220.236.54 7555 uc-k8s4p 15m
[mizuame@k8s-master-1:~$ kubectl get gameserver -n game
NAME STATE ADDRESS PORT NODE AGE
ue5-gameserver-fleet-z2jhh-5vbqv Ready 163.220.236.54 7398 uc-k8s4p 64s
ue5-gameserver-fleet-z2jhh-hksj7 Ready 163.220.236.54 7555 uc-k8s4p 16m
mizuame@k8s-master-1:~$ █
```

[2025-11-13 10:14:01] NodeFailover phase changed: Recovering → Draining

[2025-11-13 10:14:01] Condition set: GameServersDrained=True

[2025-11-13 10:14:01] Started deletion of temporary node

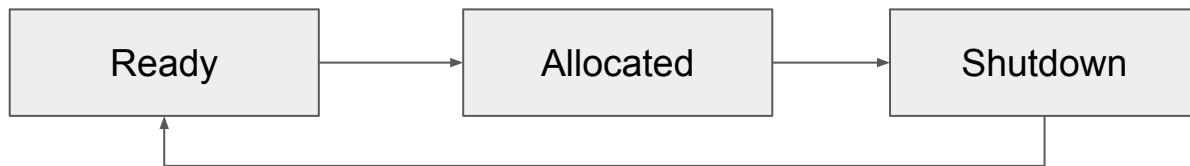
gcp-temp-uc-k8s4p-1763028338

[2025-11-13 10:14:36] Deleted node gcp-temp-uc-k8s4p-1763028338

[2025-11-13 10:14:36] Deleting GCP instance gcp-temp-uc-k8s4p-1763028338



# Agonesの状態遷移



正確にはScheduled, Requested, Startingもある

- できる限り既存のセッションは落としたくないので、すでにAllocatedされたpodが一時VMに立っている間は、VMの削除を実施しない。
- ReadyのPodは全削除
- 全てのVMのPodが消えるか、最大許容時間を超えるとVM自体の削除を実施



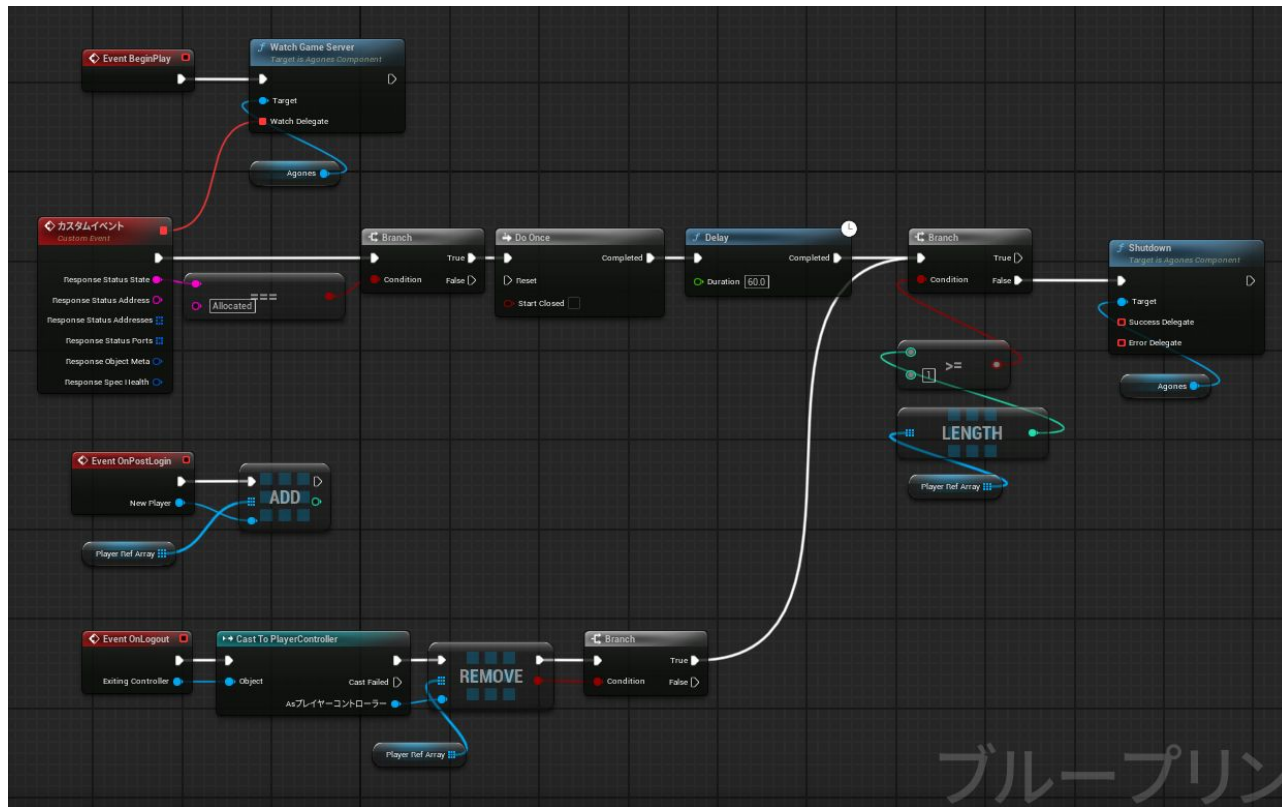
# Agonesへのシャットダウン通知実装

```
252 USTRUCT(BlueprintType)
253 struct FGameServerResponse
254 {
255 GENERATED_BODY()
256
257 UPROPERTY(BlueprintReadOnly, Category="Agones")
258 FStatus Status;
259
260 UPROPERTY(BlueprintReadOnly, Category="Agones")
261 FObjectMeta ObjectMeta;
262
263 UPROPERTY(BlueprintReadOnly, Category="Agones")
264 FSpec Spec;
265
266 FGameServerResponse()
267 {
268 }
269
270 explicit FGameServerResponse(const TSharedPtr<FJsonObject> JsonObject)
271 {
272 const TSharedPtr<FJsonObject>* ObjectMetaJsonObject;
273 if (JsonObject->TryGetObjectField(TEXT("object_meta"), ObjectMetaJsonObject))
274 {
275 ObjectMeta = FObjectMeta(*ObjectMetaJsonObject);
276 }
277 const TSharedPtr<FJsonObject>* SpecJsonObject;
278 if (JsonObject->TryGetObjectField(TEXT("spec"), SpecJsonObject))
279 {
280 Spec = FSpec(*SpecJsonObject);
281 }
282 const TSharedPtr<FJsonObject>* StatusJsonObject;
283 if (JsonObject->TryGetObjectField(TEXT("status"), StatusJsonObject))
284 {
285 Status = FStatus(*StatusJsonObject);
286 }
287 }
288 };
289
290 USTRUCT(BlueprintType)
```

```
38
39 // GameState is the state for the GameServer
40 type GameState string
41
42 const (
43 // GameStatePortAllocation is for when a dynamically allocating GameServer
44 // is being created, an open port needs to be allocated
45 GameStatePortAllocation GameState = "PortAllocation"
46 // GameStateCreating is before the Pod for the GameServer is being created
47 GameStateCreating GameState = "Creating"
48 // GameStateStarting is for when the Pods for the GameServer are being
49 // created but are not yet Scheduled
50 GameStateStarting GameState = "Starting"
51 // GameStateScheduled is for when we have determined that the Pod has been
52 // scheduled in the cluster -- basically, we have a NodeName
53 GameStateScheduled GameState = "Scheduled"
54 // GameStateRequestReady is when the GameServer has declared that it is ready
55 GameStateRequestReady GameState = "RequestReady"
56 // GameStateReady is when a GameServer is ready to take connections
57 // from Game clients
58 GameStateReady GameState = "Ready"
59 // GameStateShutdown is when the GameServer has shutdown and everything needs to be
60 // deleted from the cluster
61 GameStateShutdown GameState = "Shutdown"
62 // GameStateError is when something has gone wrong with the Gameserver and
63 // it cannot be resolved
64 GameStateError GameState = "Error"
65 // GameStateUnhealthy is when the GameServer has failed its health checks
66 GameStateUnhealthy GameState = "Unhealthy"
67 // GameStateReserved is for when a GameServer is reserved and therefore can be allocated by
68 GameStateReserved GameState = "Reserved"
69 // GameStateAllocated is when the GameServer has been allocated to a session
70 GameStateAllocated GameState = "Allocated"
71)
72
```



# Agonesへのシャットダウン通知実装





## Agonesへのシャットダウン通知実装

- AgonesSDKを使用すると、WatchGameServer()でデリゲートを取れる
- デリゲートは状態遷移が発生すると発火する。
- Allocatedが帰ってきた場合、ゲームサーバー側はクライアントが割り当てられたことを知ることができる。
- UEの場合、ゲームモードのログアウト関数の後などに、プレイヤーコントローラーの管理処理を書き、この人数が0以下になったらSDKのShutdown関数を呼ぶ
- Shutdown関数はAgonesに対し回収依頼を発行し、自動でpodがクリーンアップされる(UE自身が直接シャットダウンするわけではない)



## その他

```
pv > ! csi-s3-storageclass.yaml
1 kind: StorageClass
2 apiVersion: storage.k8s.io/v1
3 metadata:
4 name: csi-s3
5 provisioner: ru.yandex.s3.csi
6 parameters:
7 * mounter: geeseefs
8 options: "--memory-limit 1000 --dir-mode 0777 --file-mode 0666"
9 bucket: k8s
10 csi.storage.k8s.io/provisioner-secret-name: csi-s3-secret
11 csi.storage.k8s.io/provisioner-secret-namespace: kube-system
12 csi.storage.k8s.io/controller-publish-secret-name: csi-s3-secret
13 csi.storage.k8s.io/controller-publish-secret-namespace: kube-system
14 csi.storage.k8s.io/node-stage-secret-name: csi-s3-secret
15 csi.storage.k8s.io/node-stage-secret-namespace: kube-system
16 csi.storage.k8s.io/node-publish-secret-name: csi-s3-secret
17 csi.storage.k8s.io/node-publish-secret-namespace: kube-system
18 reclaimPolicy: Retain
19 volumeBindingMode: Immediate
```

```
1 apiVersion: v1
2 kind: PersistentVolume
3 metadata:
4 name: ue5-game-files-s3-pv
5 spec:
6 storageClassName: csi-s3
7 capacity:
8 storage: 10Gi
9 accessModes:
10 - ReadOnlyMany
11 claimRef:
12 namespace: game
13 name: ue5-game-files-s3-pvc
14 csi:
15 driver: ru.yandex.s3.csi
16 controllerPublishSecretRef:
17 name: csi-s3-secret
18 namespace: kube-system
19 nodePublishSecretRef:
20 name: csi-s3-secret
21 namespace: kube-system
22 nodeStageSecretRef:
23 name: csi-s3-secret
24 namespace: kube-system
25 volumeAttributes:
26 capacity: 10Gi
27 mounter: geeseefs
28 options: "--memory-limit 1000 --dir-mode 0777 --file-mode 0777"
29 volumeHandle: k8s
```



- k8s-csi-s3でcloudflare R2オブジェクトストレージをPVにしている



大体ここに載っています

mizuamedesu/**ue-k8s**



1  
Contributor

0  
Issues

0  
Stars

0  
Forks



mizuamedesu/**ue-server-env**



1  
Contributor

0  
Issues

0  
Stars

0  
Forks



<https://github.com/mizuamedesu/ue-k8s>

<https://github.com/mizuamedesu/ue-server-env>



# 謝辞

間瀬bb

[https://x.com/bb\\_mase](https://x.com/bb_mase)

Ultra-Coins

<https://ultra.coins.tsukuba.ac.jp/>

登 大遊(<https://x.com/dnabori>)先生をはじめとする、グローバルIPの貸し出しなど  
を下さっているIPAの方々など