Google Cloud

さぁ始めよう。
Agones と
Google Kubernetes Engine で
構築する ゲームサーバー

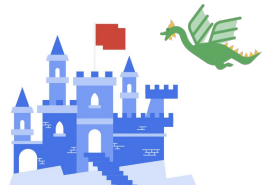
篠原 一徳

グーグル・クラウド・ジャパン合同会社
カスタマーエンジニア

スピーカー自己紹介

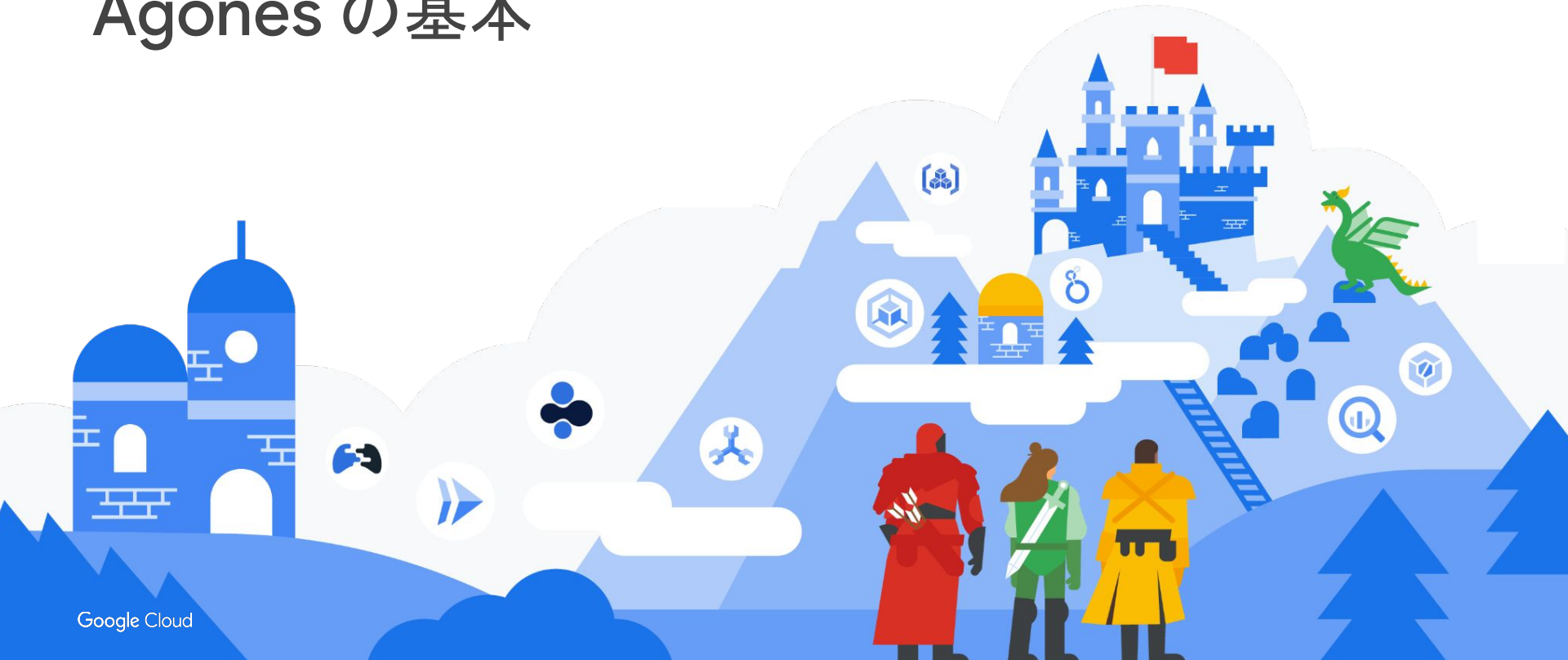
篠原 一徳
グーグル・クラウド・ジャパン合同会社
カスタマーエンジニア

ゲーム業界のお客様向けに、 GKE や Cloud Run といった
コンテナ・サーバレス関連サービスから、Agones をはじめとした Gaming
OSS の提案、アーキテクチャ設計や PoC などの
技術的なサポートを行っています。



Google Cloud

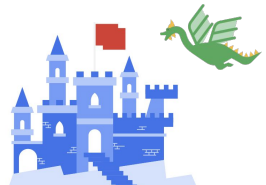
Agones の基本



Agones

Agones は **Kubernetes(K8s)** 上で
マルチプレイヤー向けの
ゲームサーバーをデプロイ / ホスト / スケーリングするための
オープンソースのプラットフォーム。

Custom Resource Definition など、
K8s の拡張機能を使って実装されている。

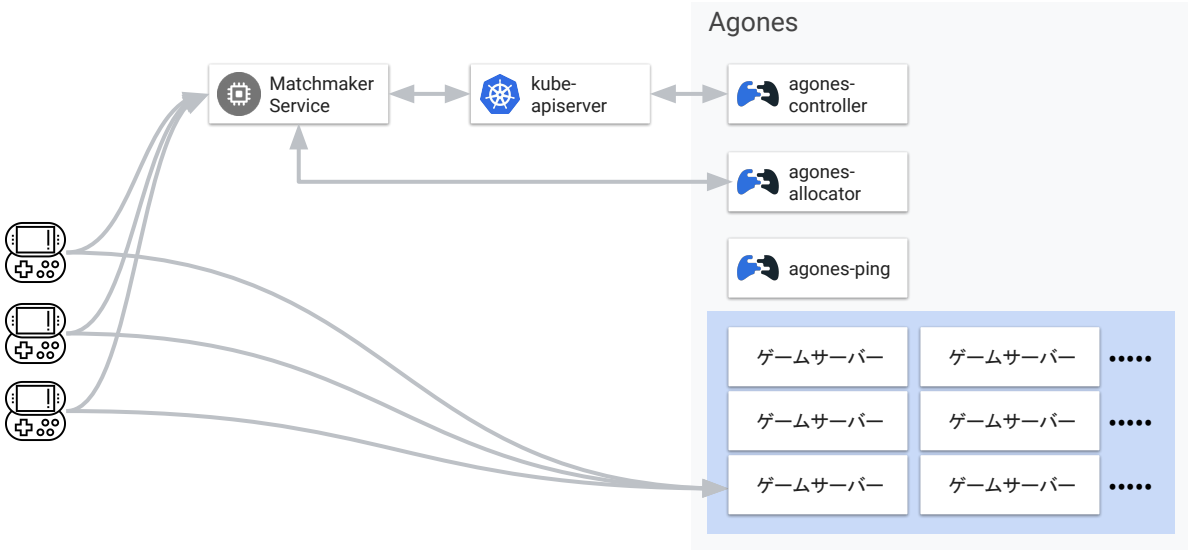


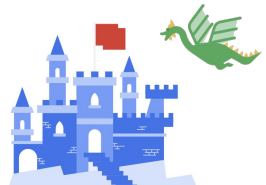
Agones のコンポーネント

agones-controller
Kubernetes のカスタム
コントローラとして動作し、Agones の
機能の大部分を提供

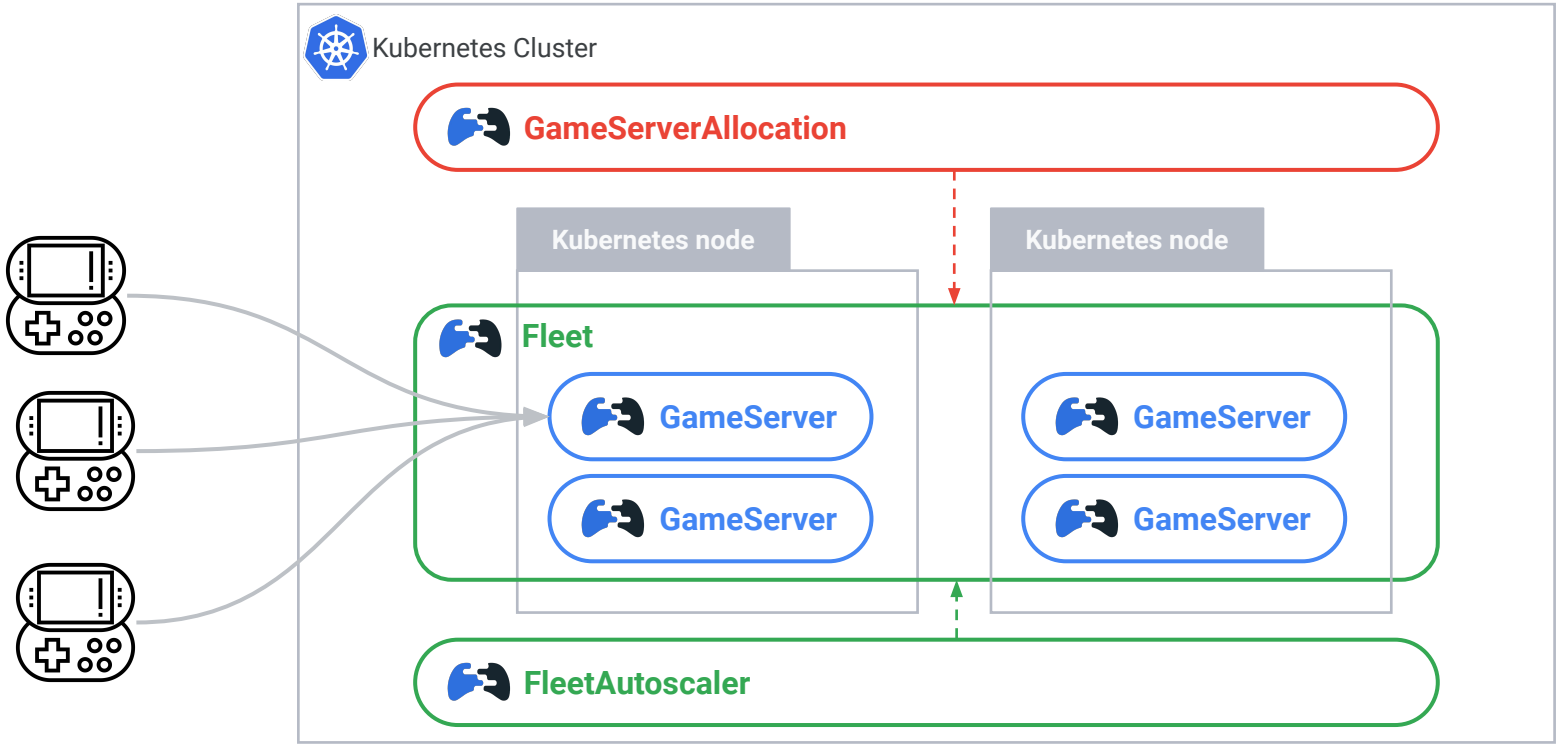
agones-allocator
ゲームサーバー のアロケーションを
行い、gRPC / REST API の
エンドポイントを提供

agones-ping
クライアントから ゲームサーバー
までのレイテンシ測定に利用



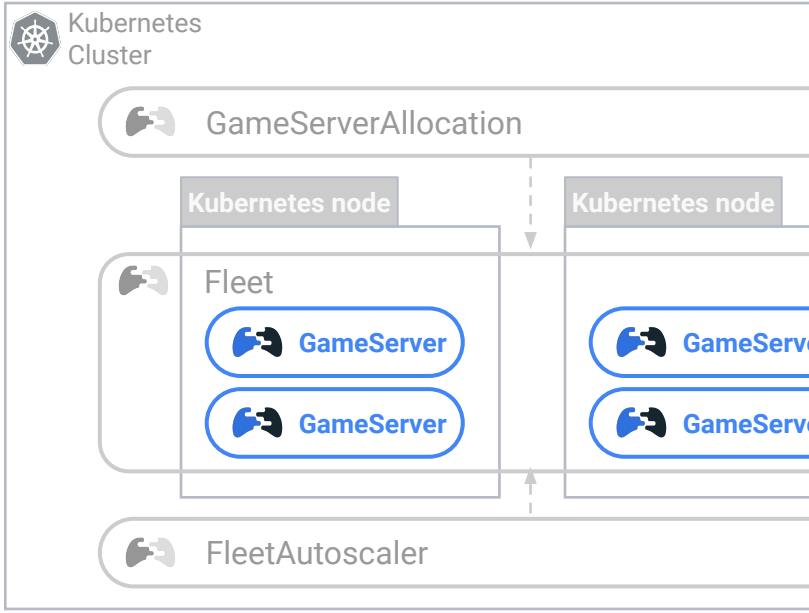


Agones の主な API リソース (K8s の 拡張リソース)

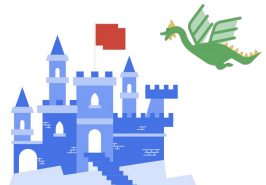


GameServer

```yaml
apiVersion: "agones.dev/v1"
kind: GameServer
metadata:
  generateName: "simple-game-server-"
spec:
  ports:
  - name: default
    portPolicy: Dynamic
    containerPort: 7654
  template:
    spec:
      containers:
      - name: simple-game-server
        image: gcr.io/agones-images/simple-game-server:0.12
```

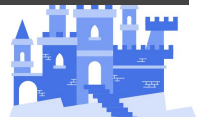


K8s の Pod に類似



GameServer の IP と Port

```
❯ kubectl get gameserver
NAME                        STATE   ADDRESS        PORT   NODE                          AGE
game-server-ftd9c-fvrtm     Ready   34.85.38.***   7020   default-pool-1121521d-j16f    72s
game-server-ftd9c-lg75w     Ready   34.85.38.***   7230   default-pool-1121521d-j16f    72s
```



補足

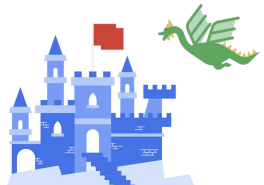
以降、2 つの類似な言葉を以下の通り使い分けています。
ただ結局、ほぼ同じ意味合いとなりますので、深く意識されなくて大丈夫です。

ゲームサーバー

主にオンライン マルチプレイなどで使用される
Dedicated Game Server 等を指す汎用的な用語

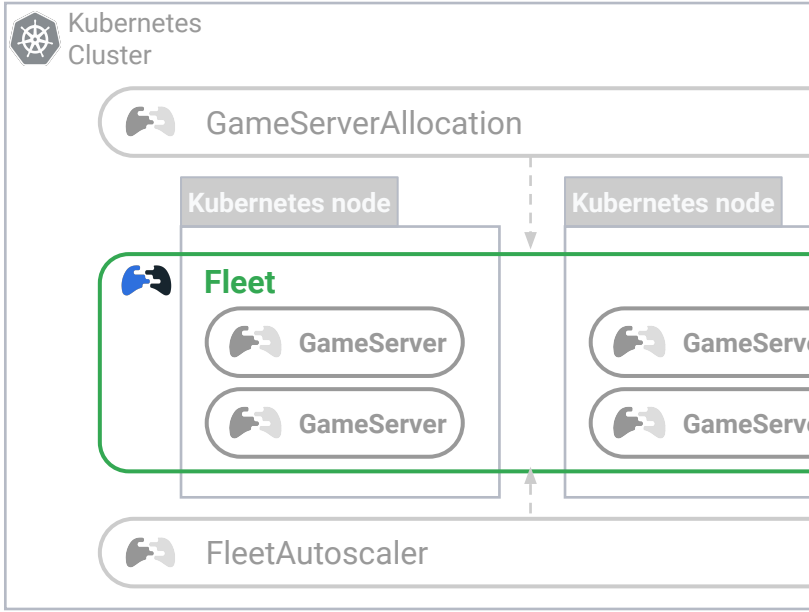
GameServer

Agones の API リソース名

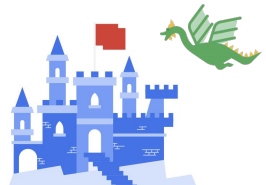


Fleet

```yaml
apiVersion: "agones.dev/v1"
kind: Fleet
metadata:
  name: simple-game-server
spec:
  replicas: 4
  template:
    spec:
      ports:
      - name: default
        portPolicy: Dynamic
        containerPort: 7654
      template:
        spec:
          containers:
          - name: simple-game-server
            image: gcr.io/agones-images/simple-game-server:0.12
```

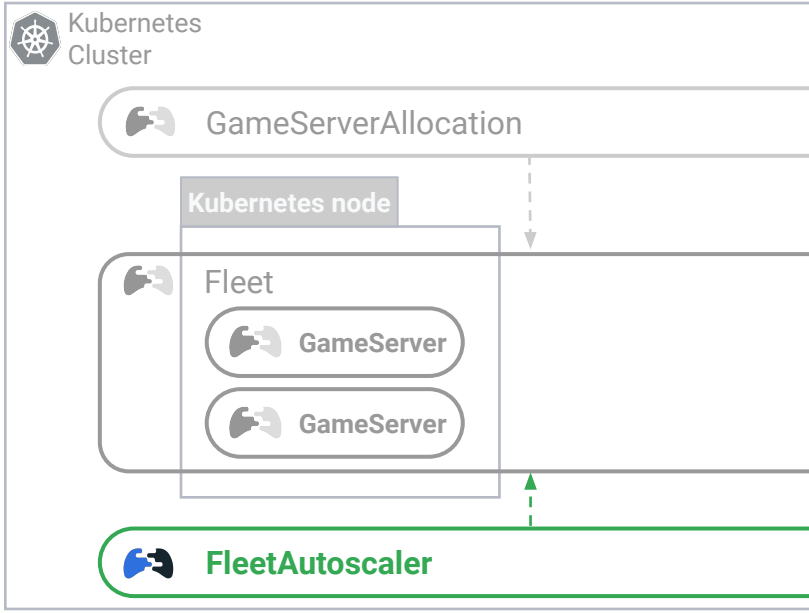


K8s の Deployment に類似

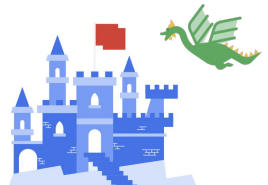


FleetAutoscaler

```yaml
apiVersion: "autoscaling.agones.dev/v1"
kind: FleetAutoscaler
metadata:
  name: simple-game-server-autoscaler
spec:
  fleetName: simple-game-server
  policy:
    type: Buffer
    buffer:
      bufferSize: 2
      minReplicas: 1
      maxReplicas: 10
```

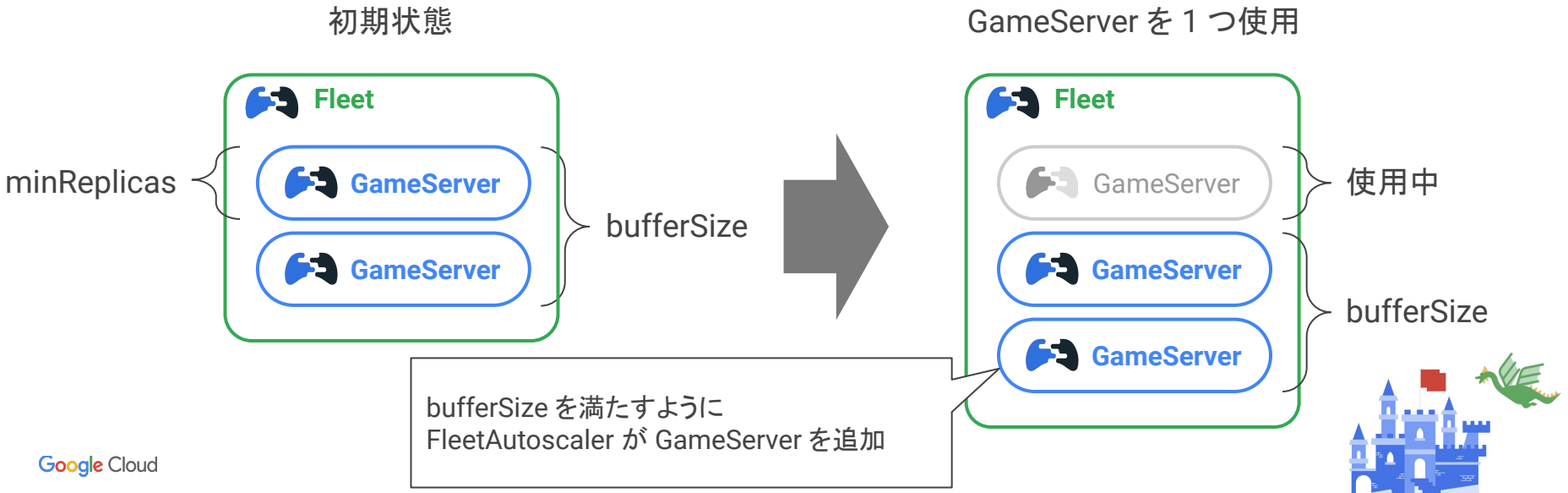


K8s の HorizontalPodAutoscaler に類似



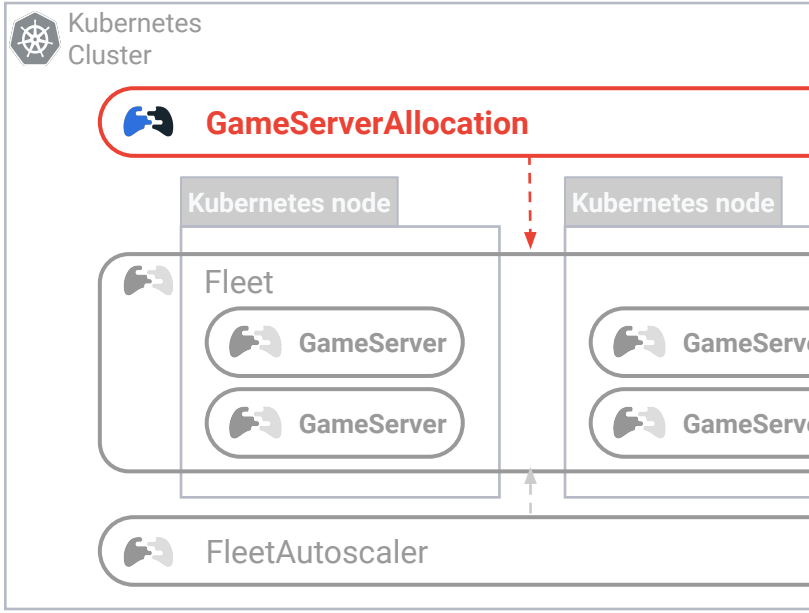
FleetAutoscaler

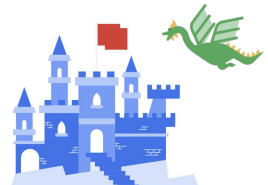
例えば、minReplicas が 1、bufferSize が 2 のとき



GameServerAllocation

```yaml
apiVersion: "allocation.agones.dev/v1"
kind: GameServerAllocation
spec:
  required:
    matchLabels:
      agones.dev/fleet: simple-game-server
```





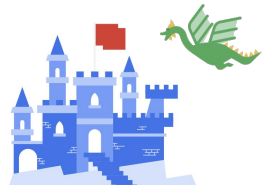
GameServerAllocation の結果

```
kind: GameServerAllocation

~~~~ 割愛 ~~~~

status:
  address: 35.193.103.***
  gameServerName: simple-game-server-c87wq-lrgv4
  nodeName: gke-agones-cluster-01-pool-1-c4b32af5-877j
  ports:
  - name: default
    port: 7097
  state: Allocated
```

クライアントからゲームサーバーへ
アクセスするための情報

- **IP Address**
- **Port 番号**

GameServer の State が
Ready -> **Allocated** へ



Google Cloud

なぜ Deployment や Pod を使うのでは駄目なのか？

- Agones の GameServer リソースは独自の State (後述) を持つことで、Pod が利用(プレイ)中なのかを判別、保護することが可能
- Kubernetes の State では Pod が利用(プレイ)中なのか判別がつかず、不慮にスケールインして削除してしまう恐れがある

GS = GameServer

Agones の環境を構築、利用する

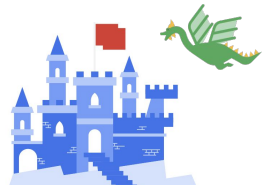
Google Cloud

Google Kubernetes Engine で Agones を利用する

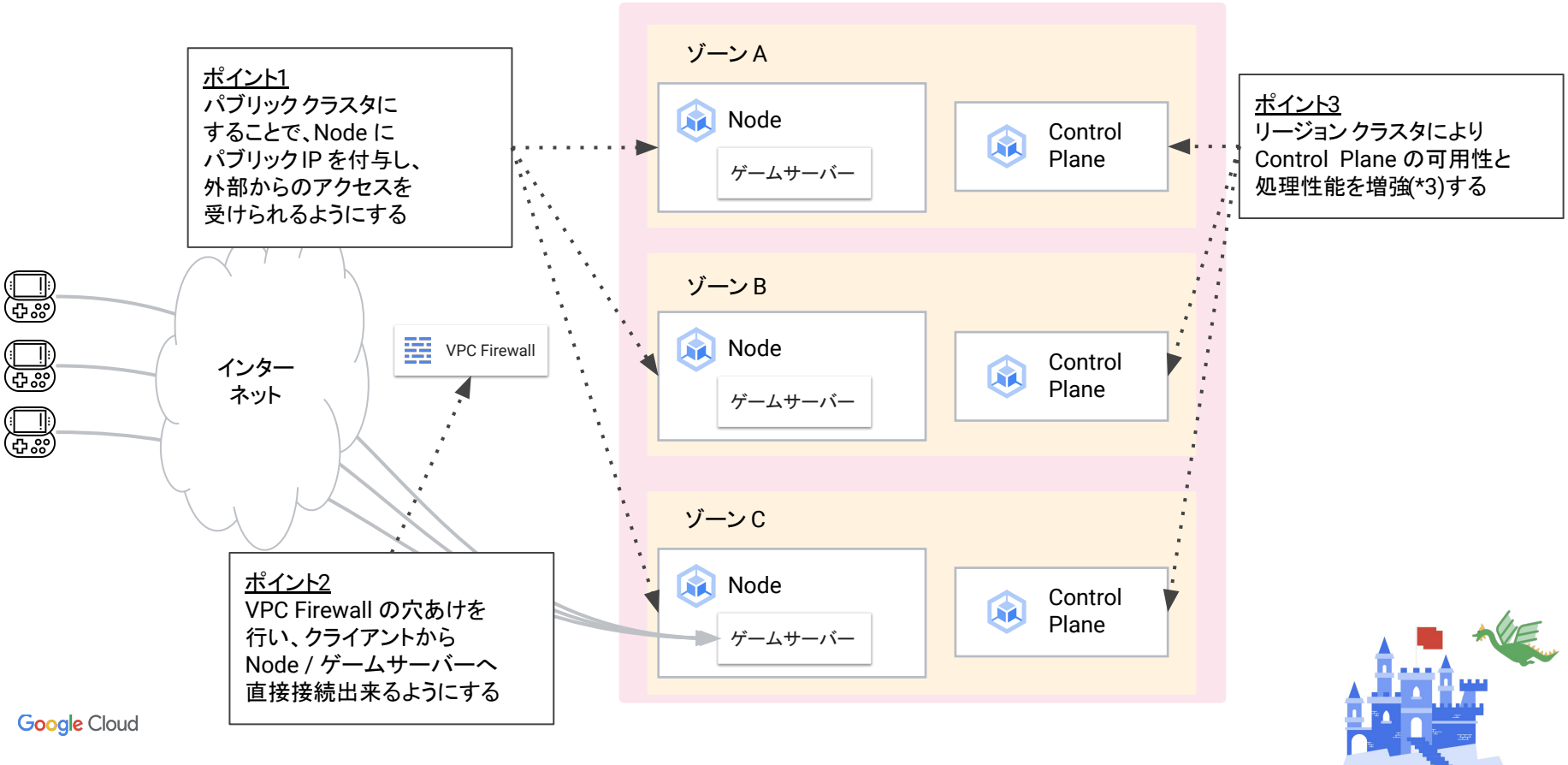
特別な理由がない限りはGoogle Kubernetes Engine (GKE) に
Agones をインストールして利用する。

推奨する GKE の主な設定は以下の通り。

- オペレーション モード: **Standard**
- 可用性タイプ: **リージョン クラスタ**
- クラスタ ネットワーク: **VPC ネイティブ クラスタ**
- バージョン: インストールする **Agones のバージョンがサポートするK8s バージョン**



GKE の構成

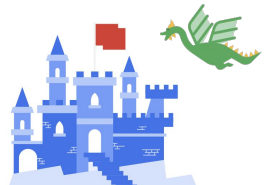

ポイント1
パブリック クラスタに
することで、Node に
パブリック IP を付与し、
外部からのアクセスを
受けられるようにする
ゾーン A
Node
ゲームサーバー
Control
Plane
ポイント3
リージョン クラスタにより
Control Plane の可用性と
処理性能を増強(*3)する
ゾーン B
Node
ゲームサーバー
Control
Plane
インター
ネット
VPC Firewall
ゾーン C
Node
ゲームサーバー
Control
Plane
ポイント2
VPC Firewall の穴あけを
行い、クライアントから
Node / ゲームサーバーへ
直接接続出来るようにする

Step 1: GKE クラスタの作成

```
$ gcloud container clusters create $CLUSTER_NAME \
  --region $REGION_NAME \
  --tags=game-server \
  --release-channel regular \
  --num-nodes  2 \
  --machine-type=e2-standard-4
```

Step 2: VPC Firewall の設定

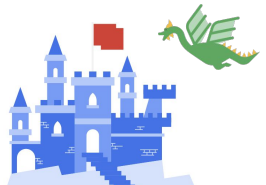
```
$ gcloud compute firewall-rules create game-server-firewall \
  --allow udp:7000-8000 \
  --target-tags game-server \
  --description "Firewall to allow ゲームサーバーudp traffic"
```



Agones のインストール

GKE へのインストール方法は3 つ存在するが、原則 **Helm** を利用することをオススメ。
インストール先の Namespace は **agones-system** をオススメ。

- **Helm** (Link)
  - 設定のカスタマイズをシンプルかつ柔軟に行える
- YAML (Link)
  - Agones レポジトリ内に予め用意されたTLS 証明書を利用
  - 設定をカスタマイズする場合、結局Helm を利用することに
- Terraform (Link)
  - GKE クラスタを1から作成し、Agones をインストール
  - まっさらな環境で、とりあえずAgones を試してみたい場合などに便利

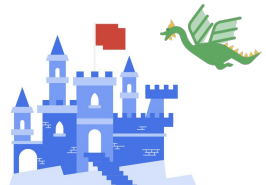


押さえておきたいインストール オプション

agones.metrics. stackdriverEnabled (Default: false)	agones-controller の メトリクスを Cloud Monitoring へエクスポート
agones.metrics. prometheusEnabled (Default: true)	agones-controller の メトリクスを /metrics, Port: 8080 で有効化
agones.ping.install (Default: true)	agones-ping を使わない場合は 無効化
agones.***.nodeSelector	controller, ping, allocator を 配置する GKE の Node Pool を指定
agones.***.logLevel (Default: “Info”)	controller, allocator を のログレベルを指定 トラシュー時に重宝

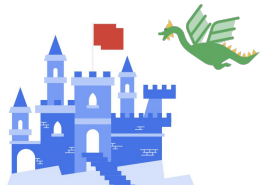
agones.allocator.service. loadBalancerIP	デフォルトでは ephemeral IP な ので固定 IP を利用
agones.allocator.service. annotations	内部 TCP/UDP LB を 使う場合は networking.gke.io/load-balanc er-type: "Internal" を指定
agones.allocator.disable MTLS (Default: true)	MTLS を使わない場合は 無効化
gameservers. namespaces (Default: “default”)	ゲームサーバーをデプロイ する Namespace を指定

他にも多くのオプションが存在します。詳しくはこちら (Link)



Step 3: Agones のインストール

```
$ helm repo add agones https://agones.dev/chart/stable

$ helm repo update

$ helm install my-release --namespace agones-system --create-namespace agones/agones
```

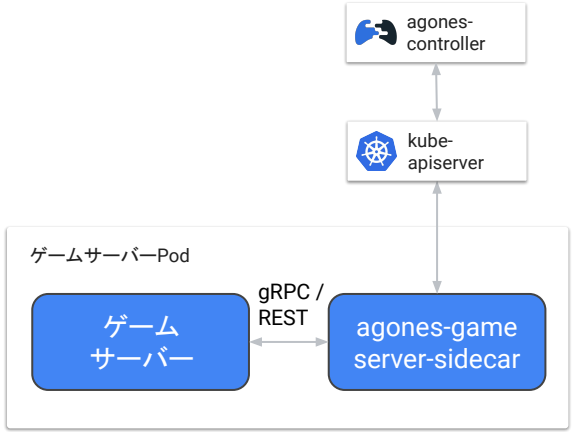


Client SDK を使ったライフサイクル管理

ゲームサーバーは Agones が提供する SDK を通じて、Agones に対して
State の変更 (READY、ALLOCATE、SHUTDOWN 等)を行う。(Link)
この際、SDK Server が サイドカーコンテナとして Agones とのやり取りを中継する。

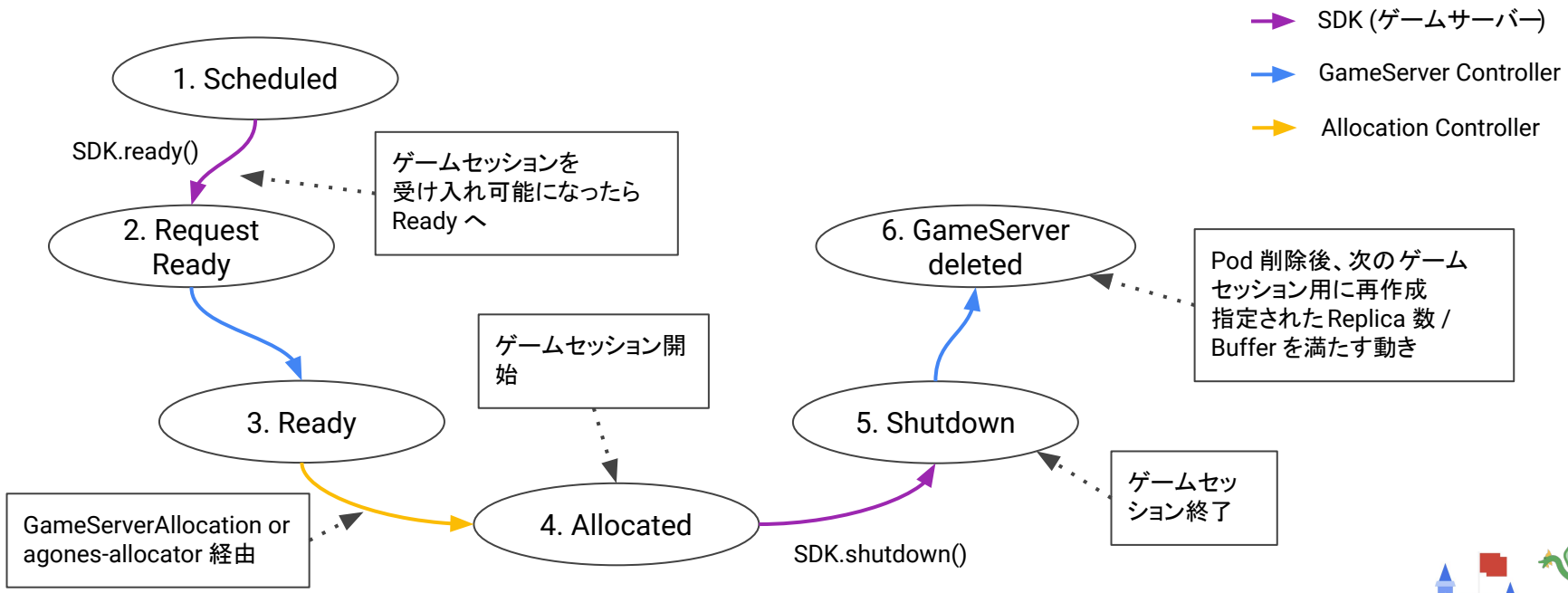
以下の言語の Client SDK がサポートされている。

- Unreal Engine
- Unity
- C++
- C#
- Node.js
- Go
- Rust
- REST



GameServer の State 遷移 (超シンプル版)

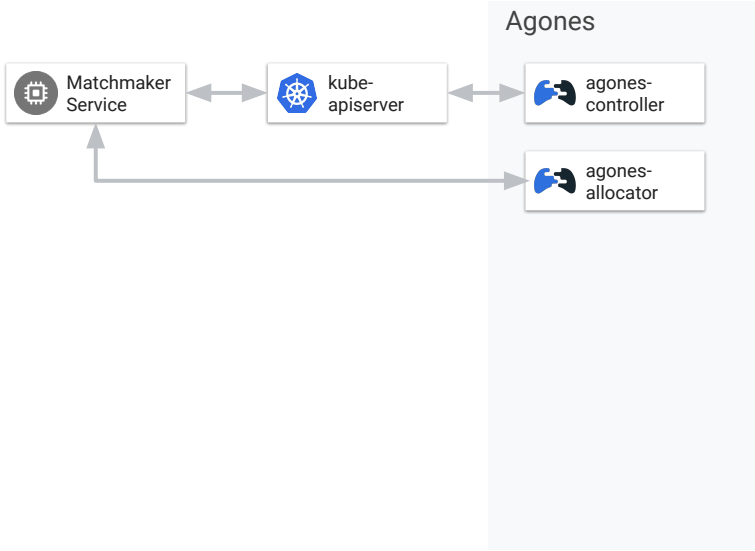
詳細は公式ドキュメントを参照のこと(Link)

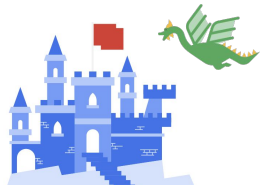


GameServerAllocation と agones-allocator

以下のような要件がある場合は、
GameServerAllocation (K8s の CRD) ではなく、
agones-allocator を使う。

- Multi-cluster allocation (Link) が必要
- Agones が動く K8s クラスタの外部から Allocation を行いたい
- K8s の RBAC ではなく、
  相互 TLS による認証を使いたい
- K8s の API ではなく、
  gRPC もしくは REST API を使用したい



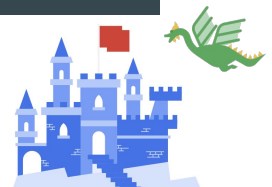


Step 4: Fleet のデプロイ (マニフェストの内容は P.9 と同じ)

```
$ kubectl apply -f fleet.yaml
```

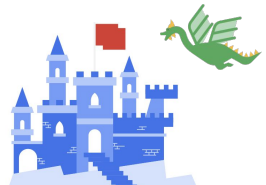
Step 5: GameServerAllocation を実施 (マニフェストの内容は P.12 と同じ)

```
$ kubectl create -f allocation.yaml -o yaml


# 一部抜粋
status:
  address: 34.146.84.***
  gameServerName: simple-game-server-4lwj7-2ll57
  nodeName: gke-agones01-default-pool-de882816-1s7l
  ports:
  - name: default
    port: 7227
  state: Allocated
```

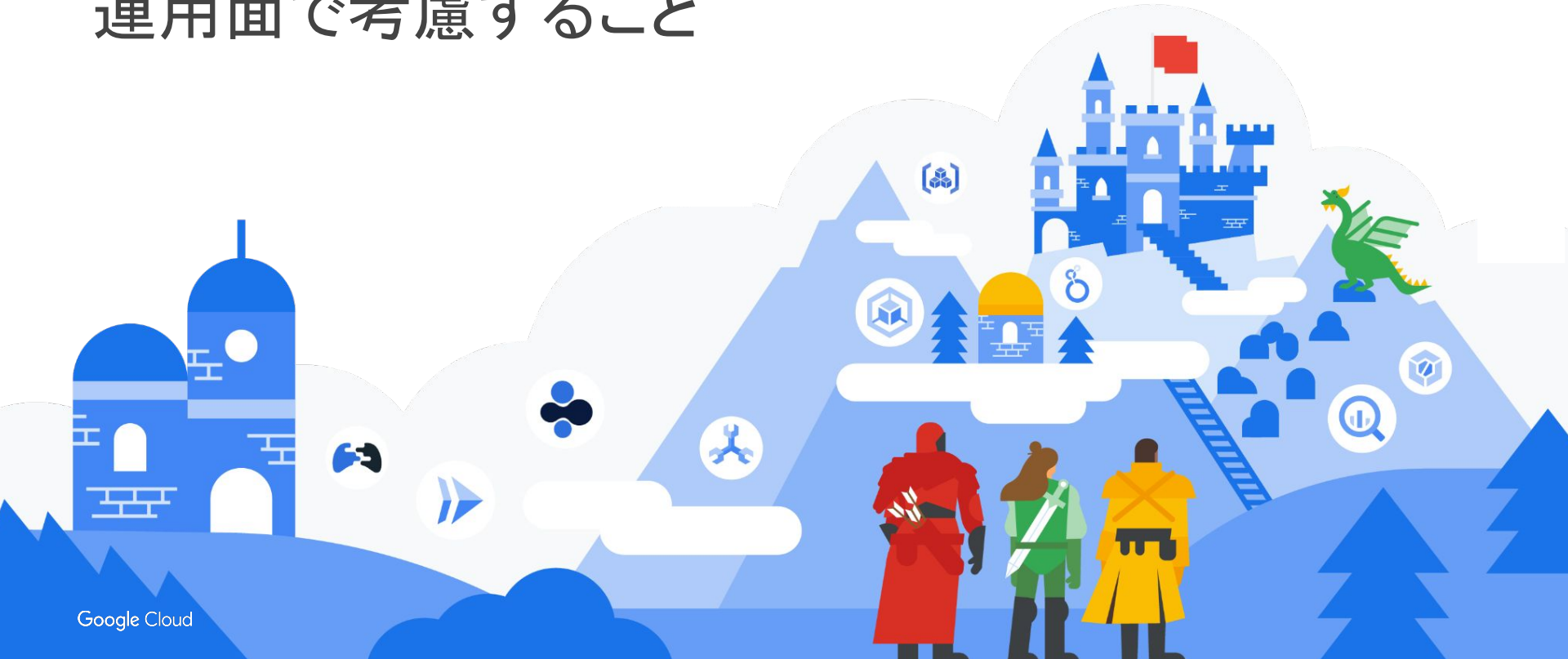


Step 6: サンプルのゲームサーバーと通信

```
$ nc -u 34.146.84.*** 7227
GAMESERVER
NAME: simple-game-server-4lwj7-2ll57

READY  # GameServer の State を Ready へ変更
ACK: READY

ALLOCATE # GameServer の State を Allocated へ変更
ACK: ALLOCATE
```



運用面で考慮すること

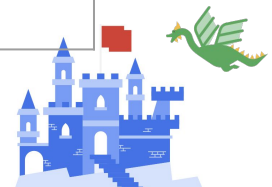


Agones と GKE のライフサイクル管理

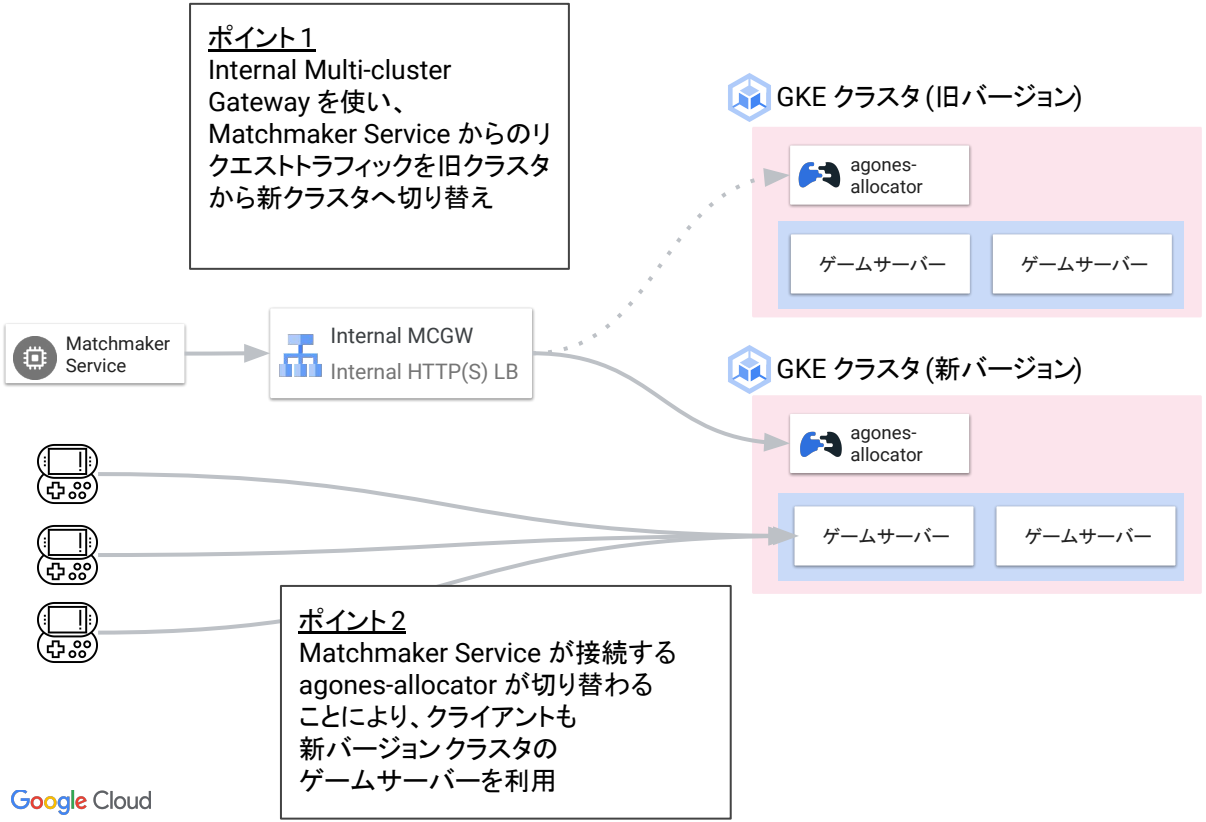
現状、**各 Agones バージョンにつき、**
1つの Kubernetes バージョンのみをサポートする。

Agones の最新バージョンは概ね、GKE の **Regular Channel** でサポートされる
バージョンと同期しているのでオススメ。

Agones バージョン	1.18 (Oct 12, 2021)	1.19 (Nov 23, 2021)	1.20 (Jan 14, 2022)	1.21 (Feb 15, 2022)	1.22 (April 04, 2022)	1.23 (May 10, 2022)
Agones がサポートする Kubernetes バージョン	1.20	1.21	1.21	1.21	1.21	1.22

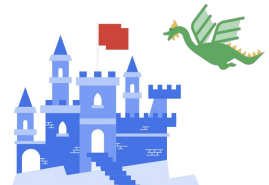


Gateway API を使ったマルチクラスタ アップグレード



Gateway API (Link)を使うことで複数クラスタ間での負荷分散が可能。

Header base routing や Traffic split など高度なルーティング機能を用いて、旧バージョン クラスタから新バージョン クラスタへスムーズな切り替えを行う。

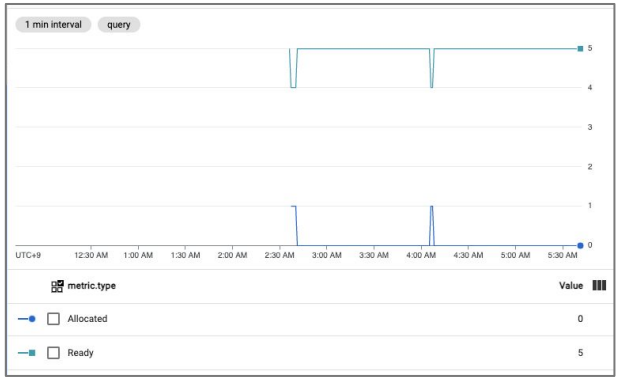


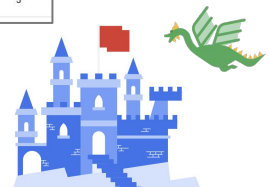
Google Cloud

メトリクス

agones-controller は OpenCensus (Link)を使いメトリクスを公開している。
Prometheus (Link) に加え Cloud Monitoring (旧名 Stackdriver、Link) の Exporter をサポート。
インストール時に有効にすればGoogle Cloud のコンソールから容易に確認できる。

agones_gameservers_count	Fleet 及び State 毎の GameServer の数
agones_nodes_count	GameServer が利用している Node と、利用していないNode の数
agones_gameserver_allocations_duration_seconds	GameServer の Allocation に掛かったレイテンシーのヒストグラム





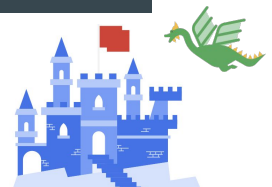
Google Cloud

ロギング

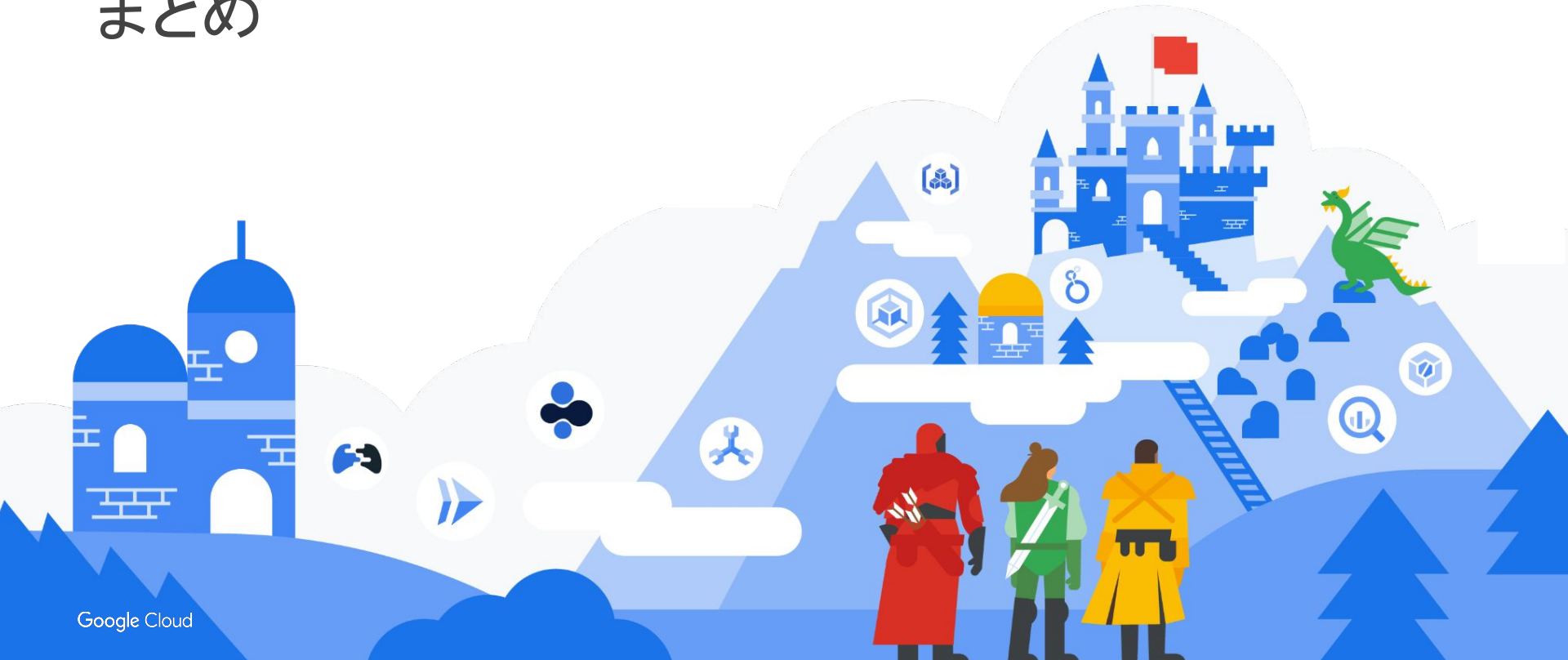
各コンポーネントから標準出力/ 標準エラー出力されるログを収集するのが基本。
GKE では特に設定をしなくてもCloud Logging (Link) に収集され、確認できる。
また kubectl logs や stern (Link)などの CLI でも確認できる。

```
$ stern simple-game-server-bgqdq-7wbpv -c agones-gameserver-sidecar

{
  "gsKey": "default/simple-game-server-bgqdq-7wbpv",
  "message": "Event(v1.ObjectReference{Kind:\"GameServer\", Namespace:\"default\",
              Name:\"simple-game-server-bgqdq-7wbpv\", UID:\"df5758d2-1870-4db5-9f5a-01e5336724db\", APIVersion:\"agones.dev/v1\",
              ResourceVersion:\"7086527\", FieldPath:\"\"}): type: Normal reason:  RequestReady SDK state change",
  "severity": "debug",
  "source": "*sdkserver.SDKServer",
  "time": "2022-06-05T21:12:02.770973229Z"
}
```

GameServer の State を Ready にするため SDK の Ready() を実行した際の SDK Server のログ



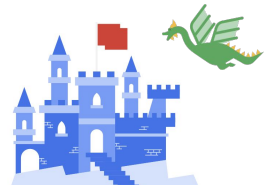
まとめ



まとめ

- 公式ドキュメント (Link)
- GitHub (Link)
- Twitter (Link)
- Slack (Link)

是非、試してみてください！



Google Cloud

Thank you