

# フルマネージド開発環境 「Cloud Workstations」 で何ができるか試してみよう

Google Cloud

アプリケーション モダナイゼーション スペシャリスト

塚越 啓介

Cloud Workstations てなに？	01
起動方法と実際の挙動	02
どんなことができるか Dive Deep	03
まとめ	04

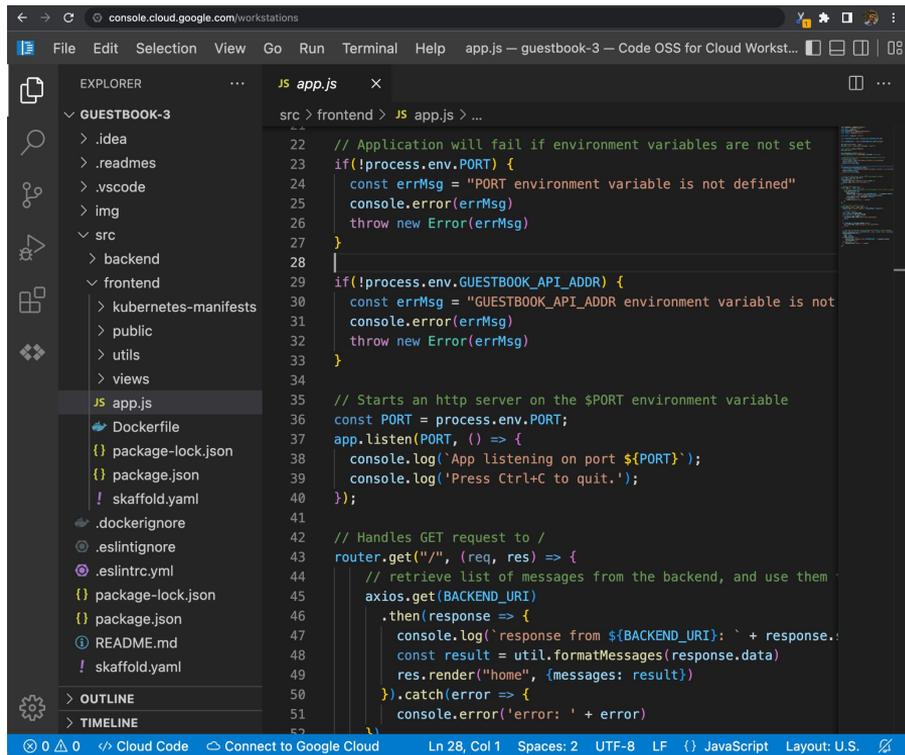
01

# Cloud Workstations

ってなに？

# Cloud Workstations

- どこからでもアクセス可能な  
オンデマンド環境
- カスタマイズ可能なツールチェーン  
(拡張機能、ライブラリ、IDE)
- チーム間で一貫した環境
- 実環境での テスト/実行
- セキュリティ ポリシー
- 管理されたベースイメージ



```
22 // Application will fail if environment variables are not set
23 if(!process.env.PORT) {
24   const errMsg = "PORT environment variable is not defined"
25   console.error(errMsg)
26   throw new Error(errMsg)
27 }
28
29 if(!process.env.GUESTBOOK_API_ADDR) {
30   const errMsg = "GUESTBOOK_API_ADDR environment variable is not
31   console.error(errMsg)
32   throw new Error(errMsg)
33 }
34
35 // Starts an http server on the $PORT environment variable
36 const PORT = process.env.PORT;
37 app.listen(PORT, () => {
38   console.log(`App listening on port ${PORT}`);
39   console.log('Press Ctrl+C to quit. ');
40 });
41
42 // Handles GET request to /
43 router.get("/", (req, res) => {
44   // retrieve list of messages from the backend, and use them
45   axios.get(BACKEND_URI)
46     .then(response => {
47       console.log(` response from ${BACKEND_URI}: ` + response.
48       const result = util.formatMessages(response.data)
49       res.render("home", {messages: result})
50     }).catch(error => {
51       console.error('error: ' + error)
```

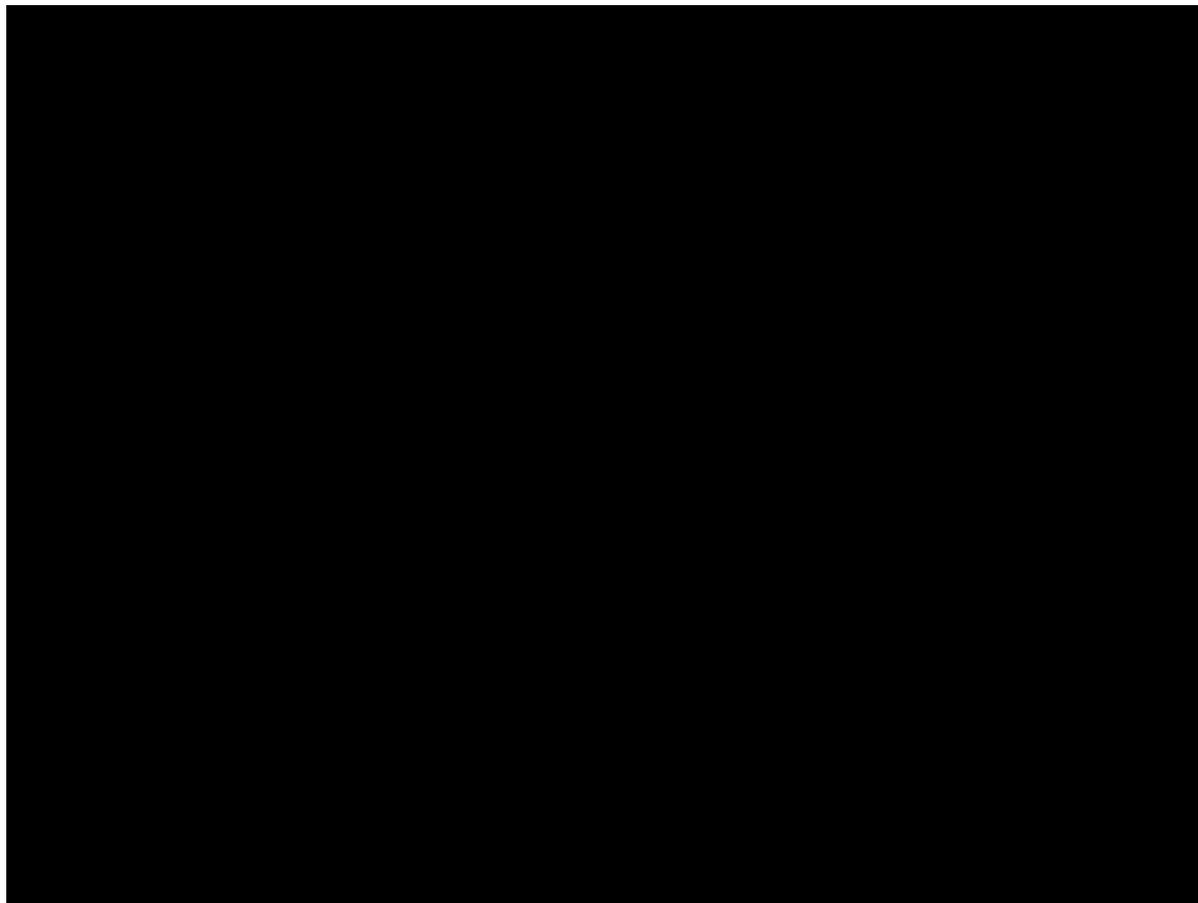
詳細はこちら <https://cloud.google.com/workstations>

02

# 起動方法と 実際の挙動

# デモ

- Cloud Workstations の動作を確認
- Cloud Workstations の起動までの流れ



Google Cloud gcsky-app@nair

Cloud Workstations **My Workstations** REVIEW CREATE REFRESH

My resources

- My Workstations

Project resources

- Workstations
- Configurations
- Clusters

### Cloud Workstations

Cloud Workstations provides managed, on-demand, development environments in the cloud. They can be accessed through a browser UI, a terminal/SSH, or from your local IDE through an SSH bridge. Get started by creating a configuration for your team. [Learn more](#)

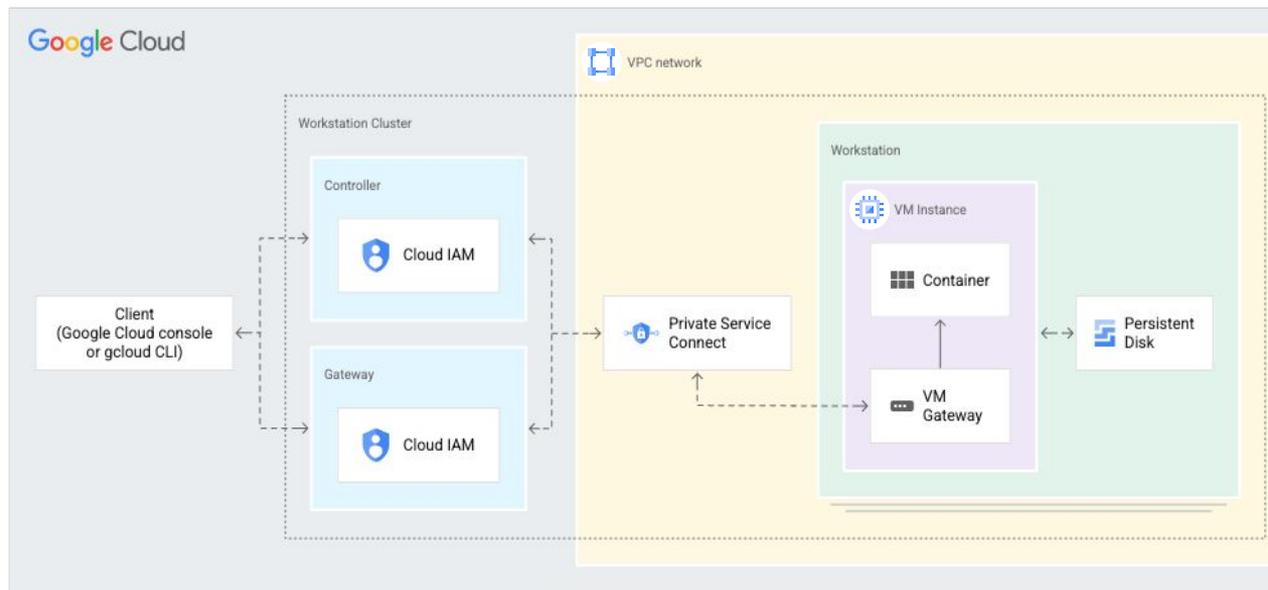
[CREATE CONFIGURATION](#) [TAKE TUTORIAL](#)

03

# Cloud Workstations の 便利な機能

# Cloud Workstations のシステム

- リソース (VM,PD)はユーザーのプロジェクト内のため、詳細な制御が可能
- プライベートネットワーク内のエンドポイントのみがクラウドワークステーションにアクセスできるようにも可能



# クラウドワークステーションのベースイメージの構造

- ベース イメージのエントリーポイント ファイルは `/google/scripts/entrypoint.sh` に設定
- 起動時に `/etc/workstation-startup.d/*` を辞書順でファイルを実行して初期化
- 現在定義されているファイルとその動作は次のとおり
  - `000_configure-docker.sh`: ワークステーション内で Docker を構成して実行
  - `010_add-user.sh`: Cloud Workstations でデフォルト ユーザーを作成
    - 永続ディスクはコンテナに動的に接続されるため、Dockerfileではなく、ワークステーションの起動時にユーザーを追加する必要あり
  - `020_start-sshd.sh`: sshd サービスを開始
  - `110_start-$IDE.sh`: イメージの IDE を起動

# コンテナイメージの カスタムの例

デフォルトユーザーをグループに追加

```
010_add-user.sh011_customize-user.sh
```

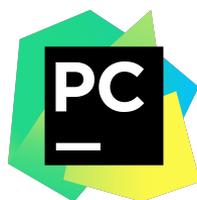
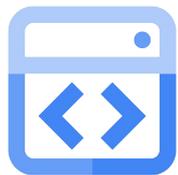
```
#!/bin/bash
# Create new group
groupadd $GROUP
# Add the user to a new group
usermod -a -G $GROUP $USERNAME
```

## Dockerfile

```
FROM
us-central1-docker.pkg.dev/cloud-workstations-images/predef
ined/code-oss:latest

COPY 011_customize-user.sh /etc/workstation-startup.d/
```

# 複数の IDE サポート + JetBrains パートナーシップ



# さまざまなインターフェースでのアクセス

## JetBrains Gateway

The screenshot shows the JetBrains Gateway IDE interface. The main editor displays the `OwnerController.java` file with the following code:

```
public String initFindForm(Map<String, Object> model) {
    model.put(k: "owner", new Owner());
    return "owners/findOwners";
}

@GetMapping("/owners")
public String processFindForm(@RequestParam(defaultValue =
    Model model) { model: size = 2

// allow parameterless GET request for /owners to return
if (owner.getLastName() == null) { owner: "[Owner@2ba677e6]
owner = (Owner@134443) "[Owner@2ba677e6 id = [null], r... View
    address = () null
    city = () null
    firstName = () null
    id = () null
    lastName = () "Ellis"
    pets = (ArrayList@134568) size = 0
    telephone = () null
```

The debugger is active, showing the execution of `processFindForm` with the following state:

- `model`: [BindingAwareModelMap@134445] size = 2
- `owner`: (Owner@134443) "[Owner@2ba677e6 id = [null], new = true, lastName = ... View
- `page`: [1, 1
- `result`: (BeanPropertyBindingResult@134444) "org.springframework.validation.BeanProp...
- `this`: (OwnerController@13075)

## VS Code via Remote-SSH

The screenshot shows the VS Code interface connected via Remote-SSH. The Explorer pane shows the file structure of a Jupyter Notebook project:

- `ch04.ipynb`
- `ch04.py`
- `README.md`
- `wine.data`
- `wine.names.txt`

The main editor displays the Jupyter Notebook content, which includes the following code cells:

```
# remove rows that contain missing values
df.dropna(axis=0)

# remove columns that contain missing values
df.dropna(axis=1)
```

The notebook also displays data tables:

A	B	C	D
0	1.0	2.0	3.0
1	6.0	6.0	6.0
2	10.0	11.0	11.0

A	B
0	1.0
1	6.0
2	10.0

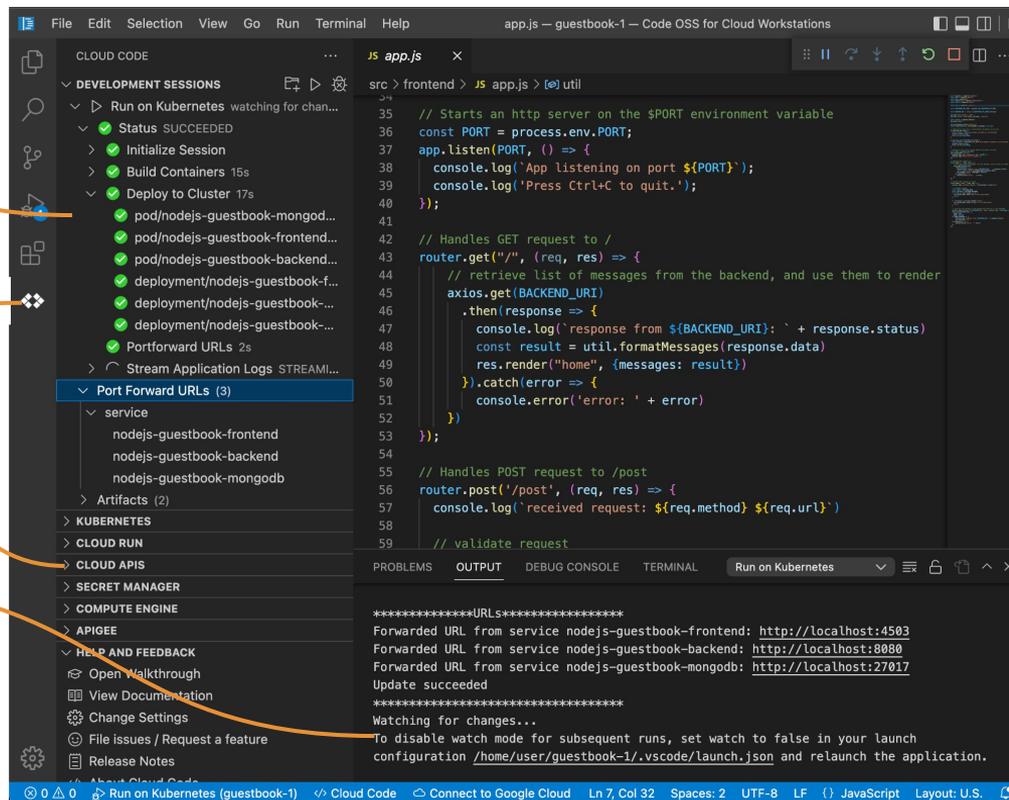
# Cloud Code とのインテグレーション

簡単に Google Cloud App を構築

IDE extensions がプリインストール

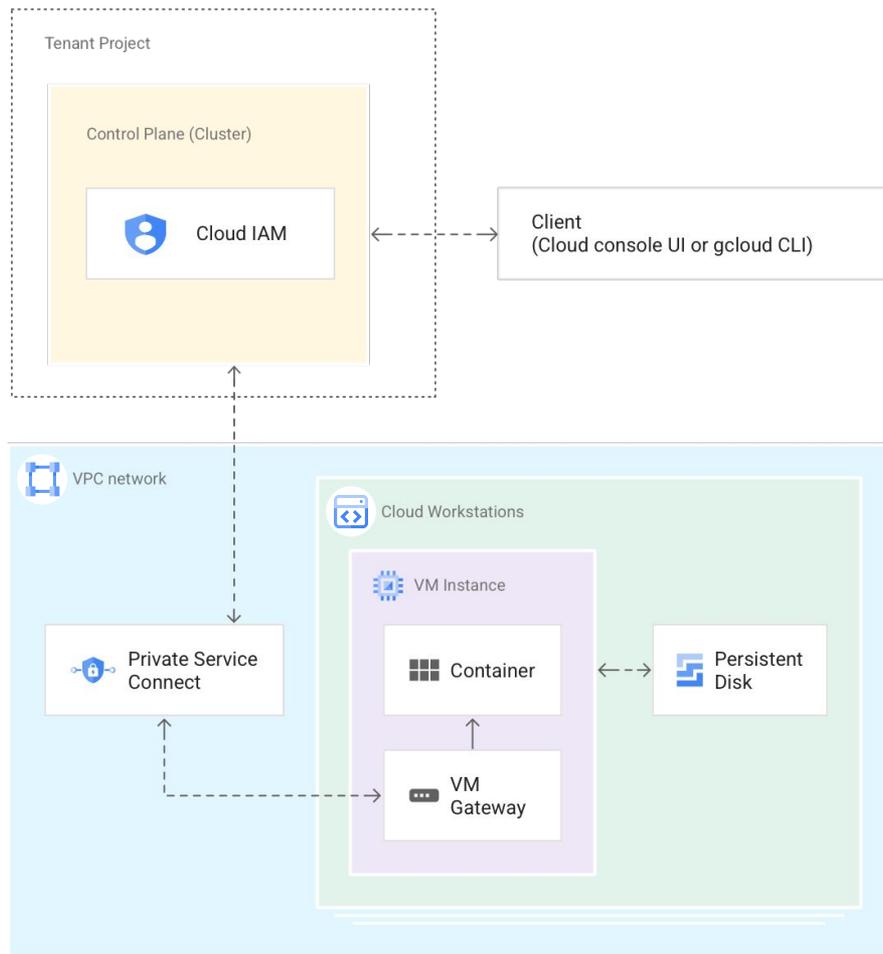
Cloud APIs と簡単に統合

code/build/test をホットリロードで



# VPC サポート

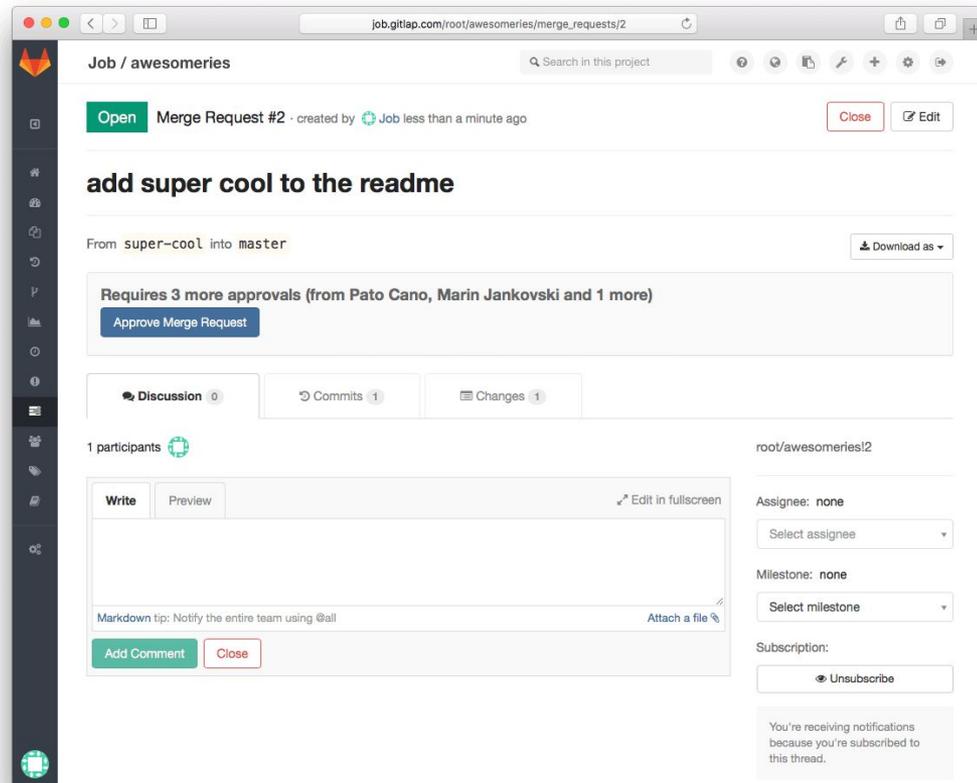
- VPC アクセス
- VPC-Service Controls
- Private Ingress/Egress のサポート
- ポリシーへの準拠



# セルフホスティング型 DevOps ツールのサポート

VPC 内から見えるセルフホスティングの  
DevOps ツールにアクセス可能

- 外部サービス
- セルフホステッド (e.g. GitLab, TeamCity, Jenkins)
- オンプレミス
- マルチクラウド



04

まとめ

# まとめ

- **ローカルの IDE に近い ユーザーエクスペリエンス**
  - デバッガ
  - 拡張機能
- **Cloud Code の統合により Google Cloud の便利な利用**
  - Cloud へのデプロイ
  - Google Cloud API 利用
- **セキュアな環境**
  - 完全にプライベートな接続
  - ローカルにデータがない



**Thank you.**