



Cloud Workstations Handson

実践 Google Cloud ハンズオン セミナー

2023 年 6 月 8 日

01

はじめるまえに

Google Cloud プロジェクトの用意

- Google Cloud のプロジェクトをまだ作成されてない方は、下記リンクの P.4 - P.11 の手順を参考に、**Google Cloud プロジェクトを作成**ください。

<https://goo.gle/startgcp1>

- クーポンコードを有効にするためには、Google Cloud プロジェクトを無料トライアルから**有料アカウントへアップグレード**する必要があります。
- アカウントをアップグレードすると、クレジットを使い切った時点、またはクレジットが期限切れになった時点で自動的に請求が始まります。

Billing Account の設定確認

- GCP コンソールのメニュー「お支払い」→「請求先アカウントを管理」
- 「マイプロジェクト」を選択し、ハンズオンで利用するプロジェクトに Billing Account が紐付いていることを確認

The screenshot shows the Google Cloud Billing Accounts page. At the top left, there is a dropdown menu labeled "組織を選択:" with a red box around it. Below the dropdown, there are two tabs: "請求先アカウント" and "マイ プロジェクト", with "マイ プロジェクト" being the active tab, indicated by a blue underline. The main table has the following columns: "名前", "ID", "請求先アカウント", "請求先アカウント ID", and "操作". There is one row of data in the table, which is heavily redacted with black bars. A vertical ellipsis icon is located at the far right end of the table row.

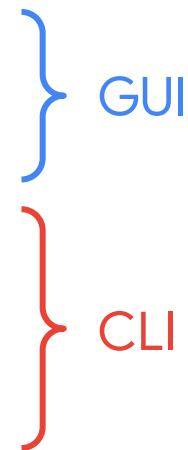
名前	ID	請求先アカウント	請求先アカウント ID	操作
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED] :

02

ハンズオンを始める

ハンズオンを開始する

1. [Google Cloud Console](#) を開く
2. Cloud Shell をアクティブにする
3. ハンズオンで利用するファイルをクローンする
4. クローンしたディレクトリに移動する
5. チュートリアルを起動する



ハンズオンを開始する(Cloud Console 上で実施)

1. [Google Cloud Console](#) を開く
2. Cloud Shell をアクティブにする



クリック

ハンズオンを開始する(Cloud Shell 上で実施)

1. ハンズオンで利用するファイルをクローンする (ホームディレクトリ直下)

```
cd ~; git clone https://github.com/GoogleCloudPlatform/gcp-getting-started-lab-jp.git
```

2. ハンズオンリソースがあるディレクトリに移動する

```
cd gcp-getting-started-lab-jp/workstations/
```

3. チュートリアルを起動する

```
teachme tutorial_ws.md
```

※ 想定のチュートリアルが開かなかった場合は、一度ブラウザのタブを閉じ

<https://console.cloud.google.com/home/dashboard> にアクセスし、

手順 2, 4, 5 を再実施してください

チュートリアルを再度起動する

1. チュートリアル リソースがあるディレクトリに移動する

```
cd ~/gcp-getting-started-lab-jp/workstations/
```

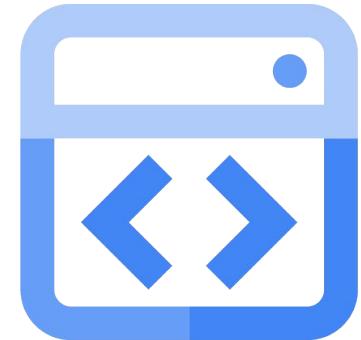
2. チュートリアルを開く

```
teachme tutorial_ws.md
```

3. ステップ 5/19 の “参考: Cloud Shell の接続が途切れてしまったときは？”まで進み
手順 3, 4 を実行する

03

Cloud Workstations 概要



Executive Summary

- Cloud Workstations は、ソフトウェア開発と関連するユースケースをカバーする、**クラウドベースの開発環境** を提供します。
- Cloud Workstationsは、Software Delivery Shield (SDS) イニシアチブの重要なコンポーネントであり、**将来のソフトウェア開発環境の基盤**を作ります。

開発チームに共通する IDE 関連の課題



開発環境のセットアップ

新/リモートメンバーの環境準備
開発メンバーが利用する
ハイスペックなマシンのコスト



セキュリティ対策/ 情報流出を防ぐための ガードレール

ローカルに保存されている
ソースコードの管理
開発者のローカル端末の
セキュリティ対策



開発者の生産性

プライベートな
ネットワーク環境で開発
ビルドに時間がかかる
アプリケーションが必要とする
資材の複雑化

Cloud Workstations:

Google Cloud ユーザーのためのマネージドな開発環境



システム管理者

- 全体の開発環境を管理
- セキュリティポリシー
- セキュアな開発環境用イメージ



開発者

- オンデマンド
- どこからでもアクセス
- 導入、設定済みの開発ツール郡
- チーム間での一貫性

The screenshot shows the Google Cloud Workstation interface. On the left is a file explorer window titled 'EXPLORER' showing a project structure for 'GUESTBOOK-3'. The structure includes '.idea', '.readmes', '.vscode', 'img', 'src' (which contains 'backend', 'frontend', 'kubernetes-manifests', 'public', 'utils', and 'views'), and files like 'Dockerfile', 'package-lock.json', 'package.json', 'skaffold.yaml', '.dockerignore', '.eslintignore', '.eslintrc.yml', 'package-lock.json', 'package.json', 'README.md', and '! skaffold.yaml'. On the right is a code editor window titled 'JS app.js' showing the following code:

```
// Application will fail if environment variables are not set
if(!process.env.PORT) {
  const errMsg = "PORT environment variable is not defined"
  console.error(errMsg)
  throw new Error(errMsg)
}

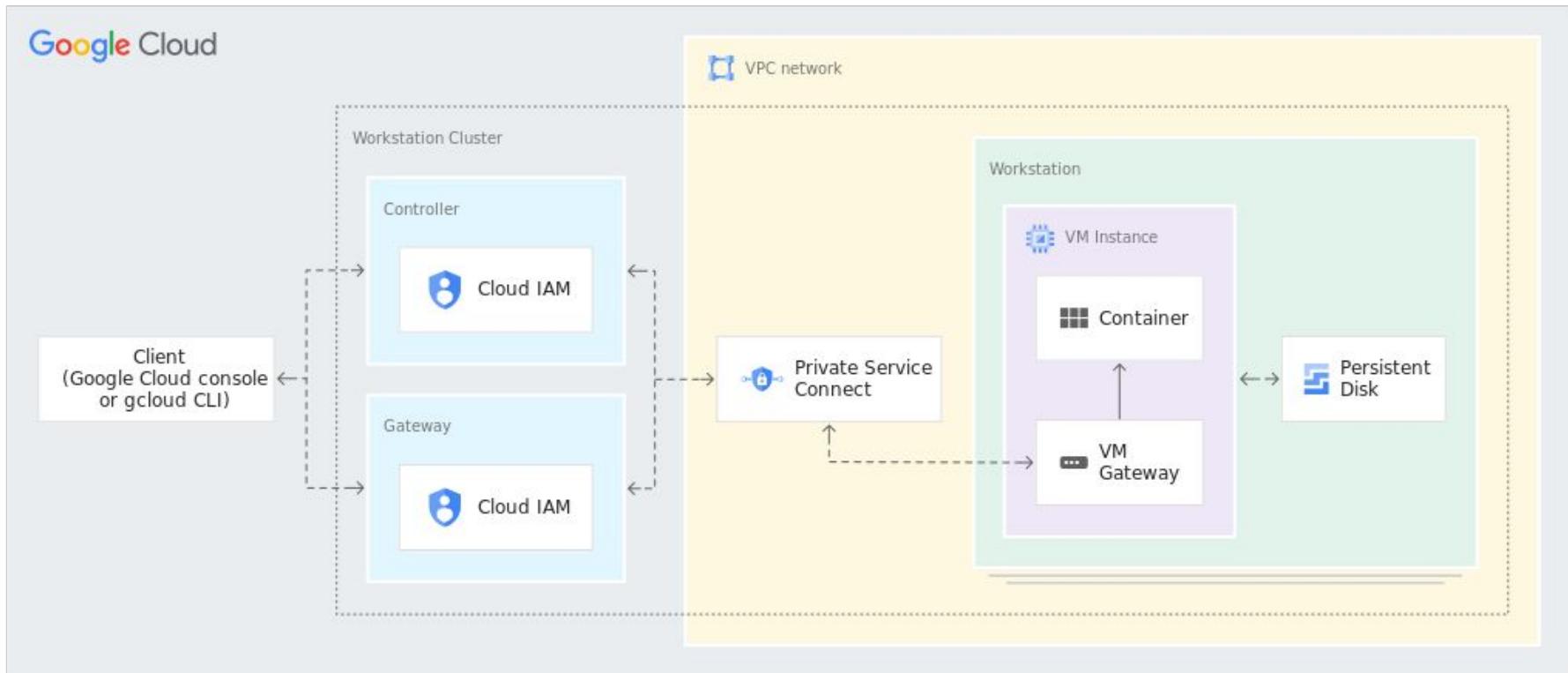
if(!process.env.GUESTBOOK_API_ADDR) {
  const errMsg = "GUESTBOOK_API_ADDR environment variable is not defined"
  console.error(errMsg)
  throw new Error(errMsg)
}

// Starts an http 'server' on the $PORT environment variable
const PORT = process.env.PORT;
app.listen(PORT, () => {
  console.log(`App listening on port ${PORT}`);
  console.log('Press Ctrl+C to quit.');
});

// Handles GET request to /
router.get("/", (req, res) => {
  // retrieve list of messages from the backend, and use them
  axios.get(BACKEND_URI)
    .then(response => {
      console.log(`response from ${BACKEND_URI}: ` + response)
      const result = util.formatMessages(response.data)
      res.render("home", {messages: result})
    })
    .catch(error => {
      console.error(`error: ${error}`)
    })
})
```

At the bottom of the interface, there are status indicators: '0 0 ▲ 0 -> Cloud Code' and 'Connect to Google Cloud'. It also shows 'Ln 28, Col 1 Spaces: 2 UTF-8 LF () JavaScript Layout: U.S.' and the 'Google Cloud' logo.

Cloud Workstations のアーキテクチャ



ポイント

満たすべき最低限の機能



エンタープライズ対応

- フルマネージド
- カスタム仮想マシン
- VPC サポート
- コンプライアンスをカバー

その他の差別化ポイント



セキュリティ ガードレール

- 厳選されたアクセスルール
- ロギング / 監査
- アップデート適用の強制
- 分離された開発環境

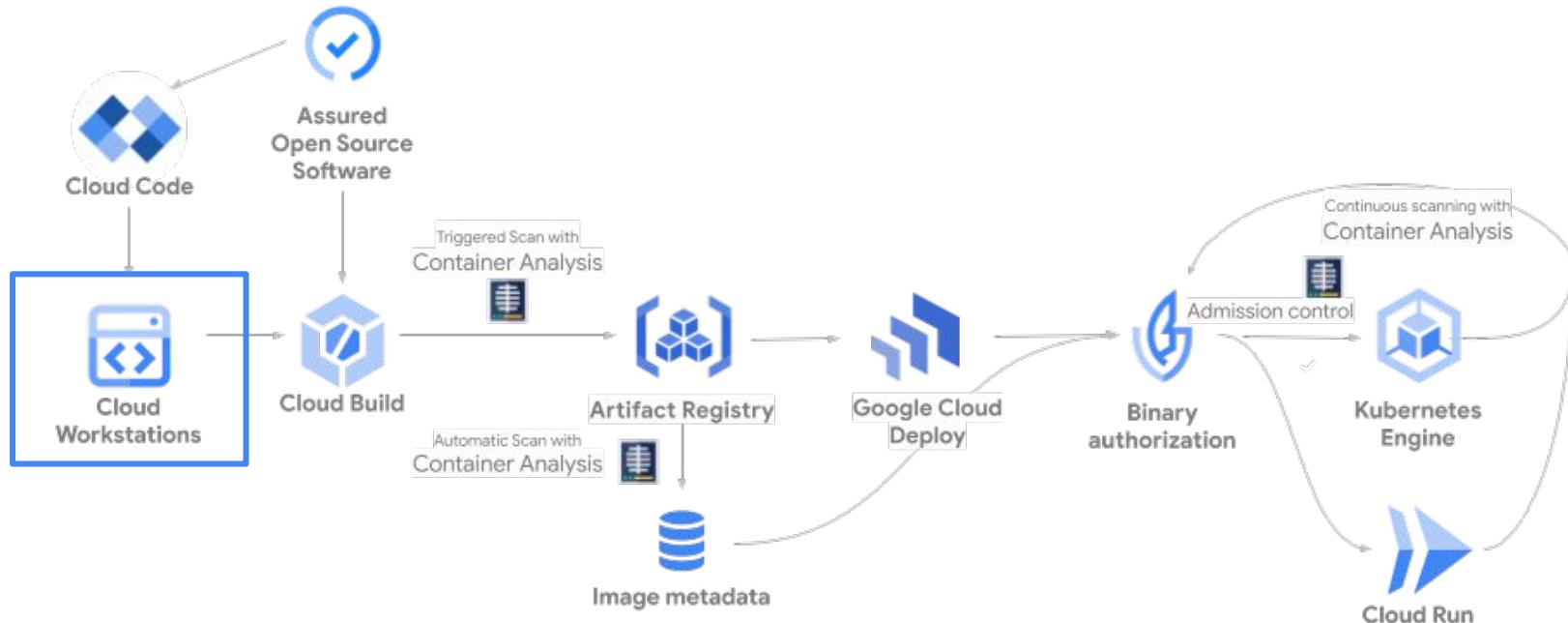


カスタマイズ可能

- カスタムイメージ
- 複数のエディタ対応 (IntelliJ, JupyterLab)
- ローカル / リモート IDE 連携
- サードパーティ DevOps ツールのサポート

Software Delivery Shield を構成するプロダクト

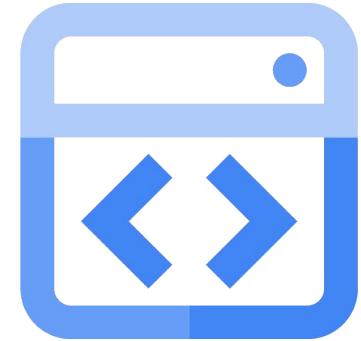
- フルマネージドでエンドツーエンドのソフトウェア サプライ チェーン セキュリティ ソリューション -



Software Delivery Shield の概要

04

Cloud Workstations Deep Dive



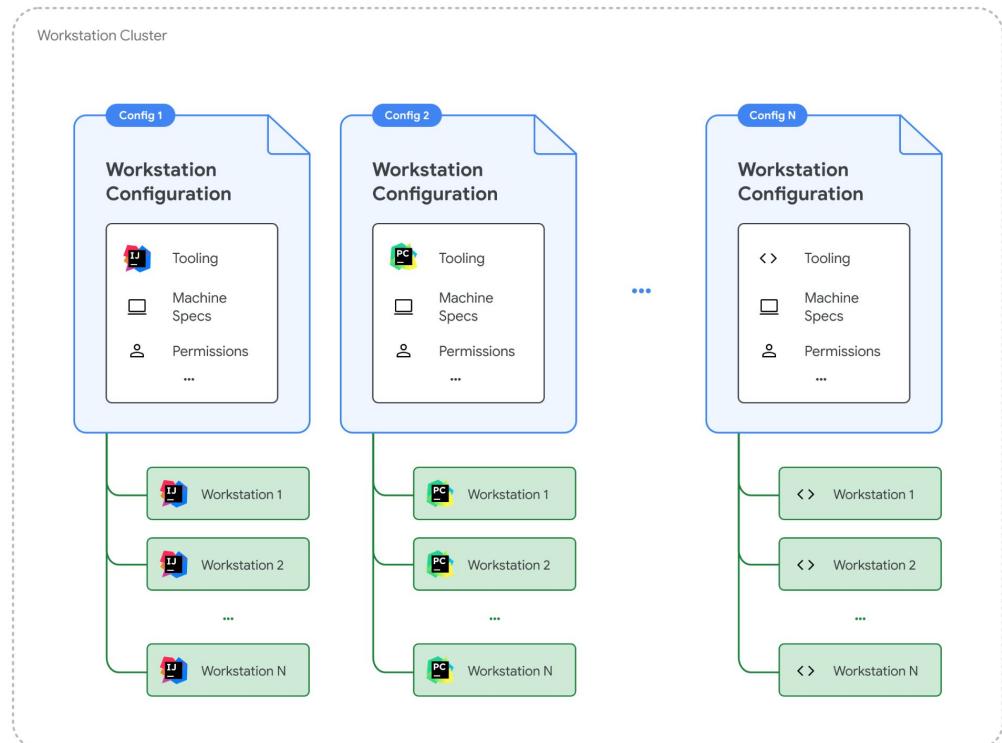
開発環境のセットアップ

数分で開発環境を用意

The screenshot shows a Google Chrome browser window with the address bar pointing to `pantheon.corp.google.com/workstations/myworkstations?project=cloud-workstations-next`. The main content area is titled "Cloud Workstations" and "My Workstations". On the left, there's a sidebar with "My resources" and "Project resources" sections, including "Workstations", "Configurations", and "Clusters". The main area features a stylized globe icon with colored dots (red, yellow, blue) and a small green dot above it. Below the icon, the text "Cloud Workstations" is displayed, followed by a description: "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 workstation." A "Learn more" link is provided. At the bottom, there are two buttons: "CREATE WORKSTATION" and "TAKE TUTORIAL". A small modal at the bottom center says "Now viewing project 'cloud-workstations-next' in organization 'hekate.joonix.net'" with a close button. The status bar at the bottom right indicates "Performance issues detected!" and "Show debug panel". The bottom right corner has the "Google Cloud" logo.

Cloud Workstations のリソース

- ワークステーション クラスタ
リージョン、VPC ネットワークに紐づく
- ワークステーション構成
コンテナイメージ、マシンスペック、
権限など。クラスタに紐づく
- ワークステーション
永続ディスク、権限など。
ワークステーション構成に紐づく



開発環境のセットアップ

一貫性のある開発環境

ワークステーション構成による共通設定

The container image defines the initial state of a workstation, such as what binaries are pre-installed and what processes are running at startup time. We provide a set of managed images with standard toolchains and some popular code editors. You can customize this further by providing your own custom image.

Managed workstation images with preinstalled code editors

Custom container image

Container image URL
gcr.io/cloud-workstations-external/java-env.latest

Service account
Compute Engine default service account

Storage settings

Disk type *
SSD

Disk size *
100 GB

Advanced container options

CREATE CANCEL

コンテナ設定による更なるカスタマイズ

```
FROM us-central1-docker.pkg.dev/cloud-workstations-images/predefined/code-oss:latest
#Install OpenJDK 17 + tools
RUN sudo apt update
RUN sudo apt install gettext-base jq httpie -y
RUN sudo apt install openjdk-17-jdk -y
#Java extension pack
RUN wget https://open-vsx.org/api/vscjava/vscode-java-pack/0.25.0/file/vscjava.vscode-jav
unzip vscjava.vscode-java-pack-0.25.0.vsix "extension/*" &&
mv extension /opt/code-oss/extensions/java-extension-pack
#Java debug
RUN wget https://open-vsx.org/api/vscjava/vscode-java-debug/0.43.0/file/vscjava.vscode-ja
unzip vscjava.vscode-java-debug-0.43.0.vsix "extension/*" &&
mv extension /opt/code-oss/extensions/java-debug
COPY ./entrypoint.sh /
RUN chmod +x /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

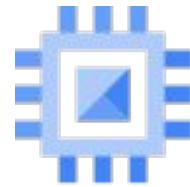
開発環境のセットアップ

柔軟なマシンスペック、GPU Preview

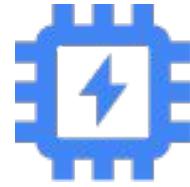
利用可能なマシンタイプ、GPU 利用可否

- **E2** (2 vCPU - 32 vCPU, 8 GB RAM - 128 GB RAM)
- **N1** (2 vCPU - 96 vCPU, 7.5 GB RAM - 360 GB RAM)
- **N2** (2 vCPU - 32 vCPU, 8 GB RAM - 128 GB RAM)
- **N2D** (2 vCPU - 32 vCPU, 8 GB RAM - 128 GB RAM)
- **Tau T2D** (60 vCPU, 240 GB RAM)
- **A2** (12 vCPU - 96 vCPU, 85 GB RAM - 60 GB RAM)

GPU 搭載化



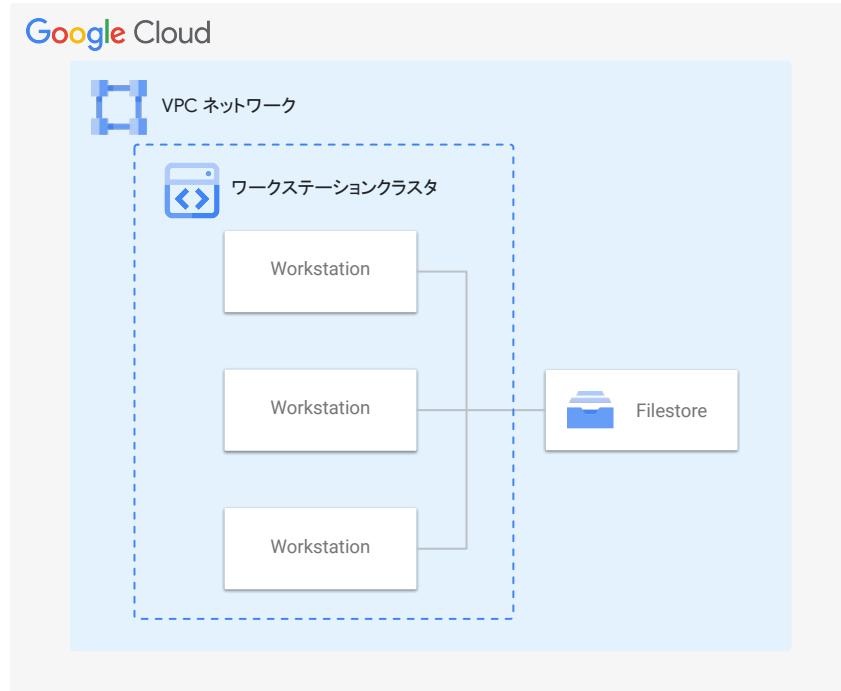
GPU 搭載化



使用可能なマシンタイプ , 使用可能な GPU

Filestore を用いたデータの共有

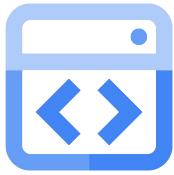
- 複数のワークステーションでデータを共有
- 複数利用者の同時書き込み
- VPC に閉じた通信



[Filestore インスタンスを Cloud Workstations にマウントする](#)

開発者の生産性向上

複数の IDE サポート + JetBrains 社とのパートナーシップ



開発者の生産性向上

複数のインターフェースからアクセス可能



JetBrains Gateway

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "petclinic" and contains modules for "samples", "petclinic", and "owner". The "owner" module has a "src" directory with "main" and "test" packages, and a "checkstyle" configuration.
- Code Editor:** The file "OwnerController.java" is open, showing code related to handling owner requests. A break point is set at line 94, and the code is annotated with various inspection markers like "Red", "Yellow", and "Green".
- Run Tab:** The application "PetClinicApplication" is running on port 8080, as indicated by the green "RUNNING" status.
- Debug Tab:** The "Evaluate expression" tool window is open, showing the current state of variables. It includes fields for "model", "owner", "page", "result", and "this".
- Bottom Navigation:** The "Debug" tab is selected, along with other tabs like "Git", "Run", "Terminal", "Profiler", "Dependencies", "Services", "Problems", "Endpoints", "Spring", "Build", and "Main".



VS Code via Remote-SSH

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

	A	B	C	D
0	1.0	2.0	3.0	4.0

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

	A	B
0	1.0	2.0
1	5.0	6.0
2	10.0	11.0

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

	A	B
0	1.0	2.0
1	5.0	6.0

PROBLEMS ① OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER: VARIABLES

user@vscode-test-3:~/machine-learning-book/ch04\$

開発者の生産性向上

Cloud Code との連携



Google Cloud にアプリをデプロイ

インストール済みの IDE エクステンション

Google Cloud API と簡単に連携

ホットリロードを使った高速な コーディング/ビルド/テストの 開発サイクル

The screenshot shows the Cloud Workstations interface with the following components:

- File Explorer:** On the left, it displays a tree view of development sessions, port forward URLs, artifacts, Kubernetes, Cloud Run, Cloud APIs, Secret Manager, Compute Engine, APIGEE, help, and feedback.
- Terminal:** At the bottom, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and RUN ON KUBERNETES.
- Code Editor:** The main area shows the file `app.js` with the following content:

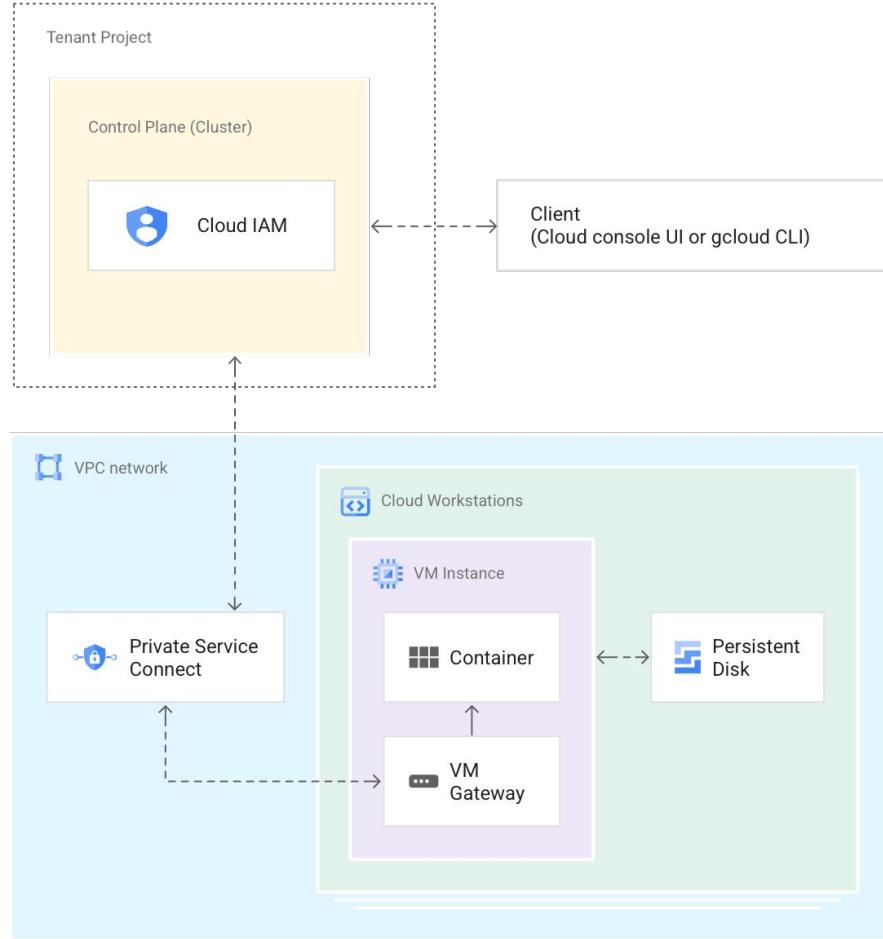
```
src.frontend > JS app.js > [e] util
35 // Starts an http server on the $PORT environment variable
36 const PORT = process.env.PORT;
37 app.listen(PORT, () => {
38   console.log(`App listening on port ${PORT}`);
39   console.log('Press Ctrl+C to quit.');
40 });
41
42 // Handles GET request to /
43 router.get("/", (req, res) => {
44   // retrieve list of messages from the backend, and use them to render
45   axios.get(BACKEND_URL)
46     .then(response => {
47       console.log(`Response from ${BACKEND_URL}: ` + response.status)
48       const result = util.formatMessages(response.data)
49       res.render("home", {messages: result})
50     }).catch(error => {
51       console.error(`Error: ${error}`)
52     })
53 });
54
55 // Handles POST request to /post
56 router.post('/post', (req, res) => {
57   console.log(`Received request: ${req.method} ${req.url}`)
58
59 // validate request
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Run on Kubernetes
```

Below the code editor, the terminal output shows:

```
*****URLS*****
Forwarded URL from service nodejs-guestbook-frontend: http://localhost:4503
Forwarded URL from service nodejs-guestbook-backend: http://localhost:8080
Forwarded URL from service nodejs-guestbook-mongodb: http://localhost:27017
Update succeeded
*****
Watching for changes...
To disable watch mode for subsequent runs, set watch to false in your launch configuration /home/user/guestbook-1/.vscode/launch.json and relaunch the application.
```

エンタープライズ対応
VPC サポート

- VPC へのアクセス
- VPC Service Controls との連携
- プライベート Ingress / Egress のサポート
- 自動的なネットワークのタグ付け
- プロジェクトの組織ポリシーに準拠



エンタープライズ対応

開発環境を集中管理

- マネージドなベースイメージ
- カスタマイズ可能な仮想マシンタイプとイメージ
- IAM 権限での制御
- イメージのアップデートを強制適用化

The screenshot shows a Google Cloud Platform interface for managing Cloud Workstations. The top navigation bar includes 'Google Cloud', 'cloud-workstations-next', a search bar, and various user and settings icons. A sidebar on the left has icons for Cloud Workstations, Compute Engine, and other services. The main area displays a table of '2 resources selected'.

Table Headers:

- Status
- Name ↑
- Location
- Cluster
- Machine type
- Created
- Last updated

Data Rows:

Status	Name	Location	Cluster	Machine type	Created	Last updated
<input type="checkbox"/> READY	cpp-env	us-central1	test-cluster	e2-standard-4	Sep 2, 2022, 7:55:04 AM	Sep 2, 2022, 7:55:52 AM
<input checked="" type="checkbox"/> READY	golang-env	us-central1	test-cluster	e2-standard-4	Sep 2, 2022, 7:54:24 AM	Sep 2, 2022, 7:54:58 AM
<input checked="" type="checkbox"/> READY	intellij-ide	us-central1	test-cluster	e2-standard-4	Sep 2, 2022, 1:44:39 PM	Sep 2, 2022, 1:45:26 PM
<input type="checkbox"/> READY	java-env	us-central1	test-cluster	e2-standard-8	Aug 31, 2022, 2:52:50 PM	Sep 3, 2022, 6:50:57 PM
<input type="checkbox"/> READY	nodejs-env	us-central1	test-cluster	e2-standard-4	Sep 2, 2022, 7:55:19 AM	Sep 2, 2022, 7:55:56 AM
<input type="checkbox"/> READY	python-env	us-central1	test-cluster	e2-standard-4	Sep 2, 2022, 7:54:44 AM	Sep 2, 2022, 7:55:20 AM

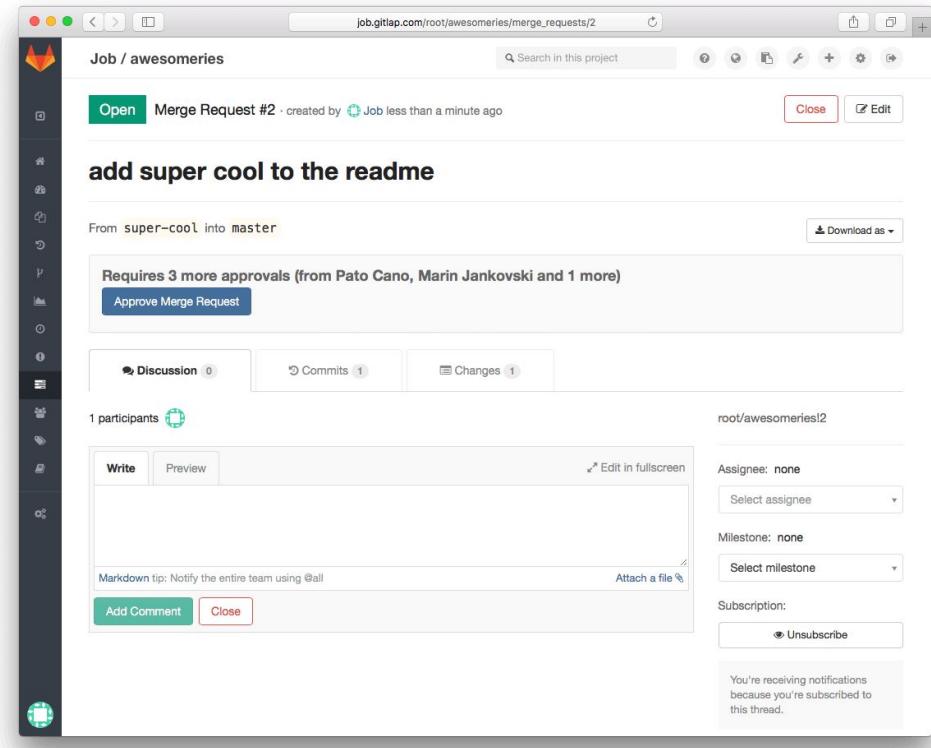
エンタープライズ対応

セルフホスト型 DevOps ツールをサポート

VPC 内から表示できるセルフホスト型

DevOps ツールとの連携

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

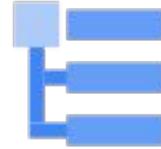


エンタープライズ対応

監査とロギング

自動で出力、取得される様々なログ

- 監査ログ
 - 誰が、いつ、どこで、何をしたか
- ワークステーション コンテナログ
 - 標準出力、標準エラーログ
- BigQuery と連携させ、ログを分析



[Cloud Workstations 監査ロギング](#)

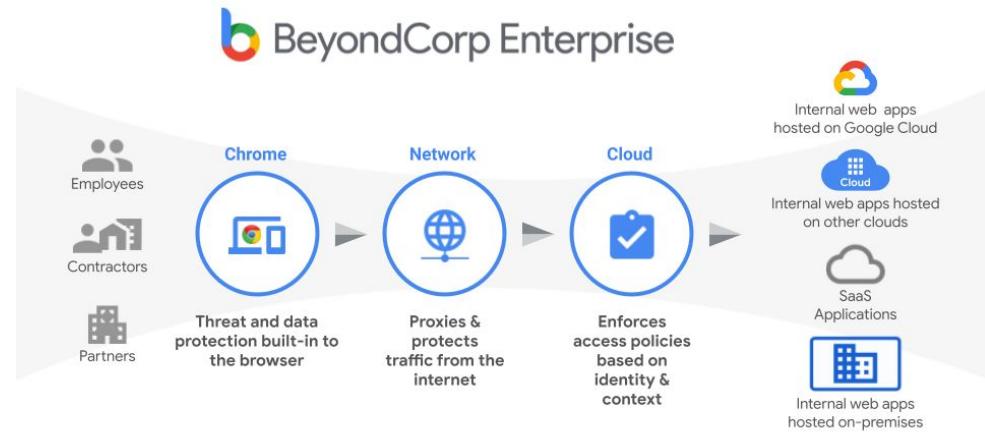
Google Cloud

エンタープライズ対応

BeyondCorp Enterprise と連携 Preview

セキュリティを高度化

- 様々な条件をもとにアクセスを制御
 - IP アドレス、デバイス、時間など
- ソースコードの漏洩を防ぐ
 - ファイルのアップロード、ダウンロード
 - コピー & ペースト



[BeyondCorp Enterprise を使用して Cloud Workstations API を保護する](#)

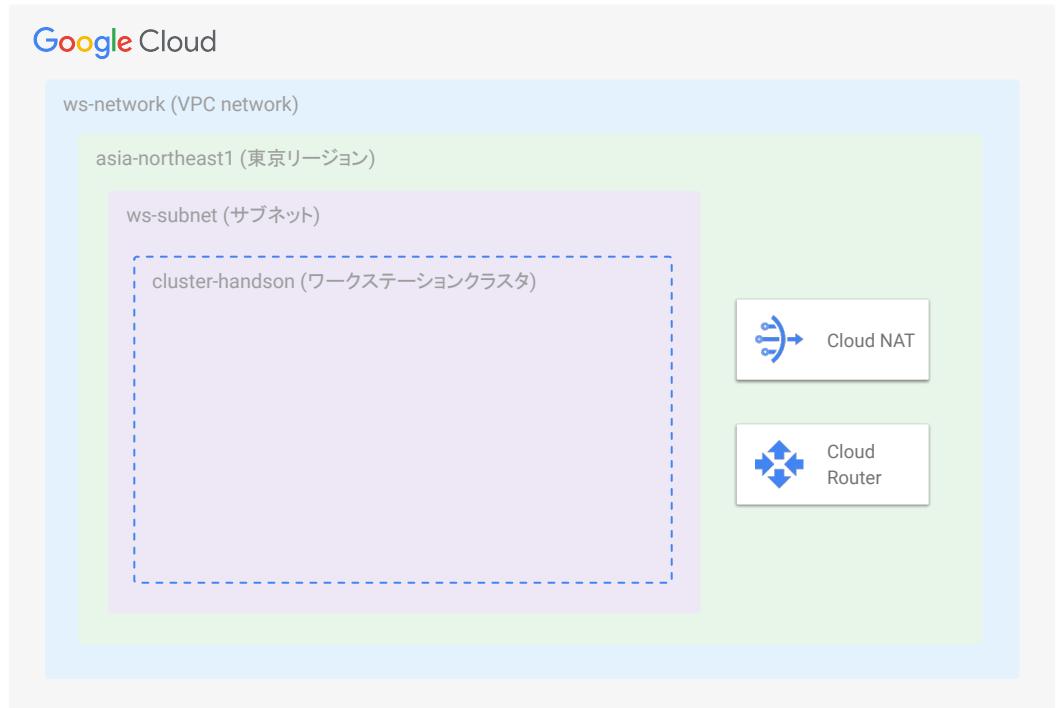
Google Cloud

05

図解ハンズオンステップ

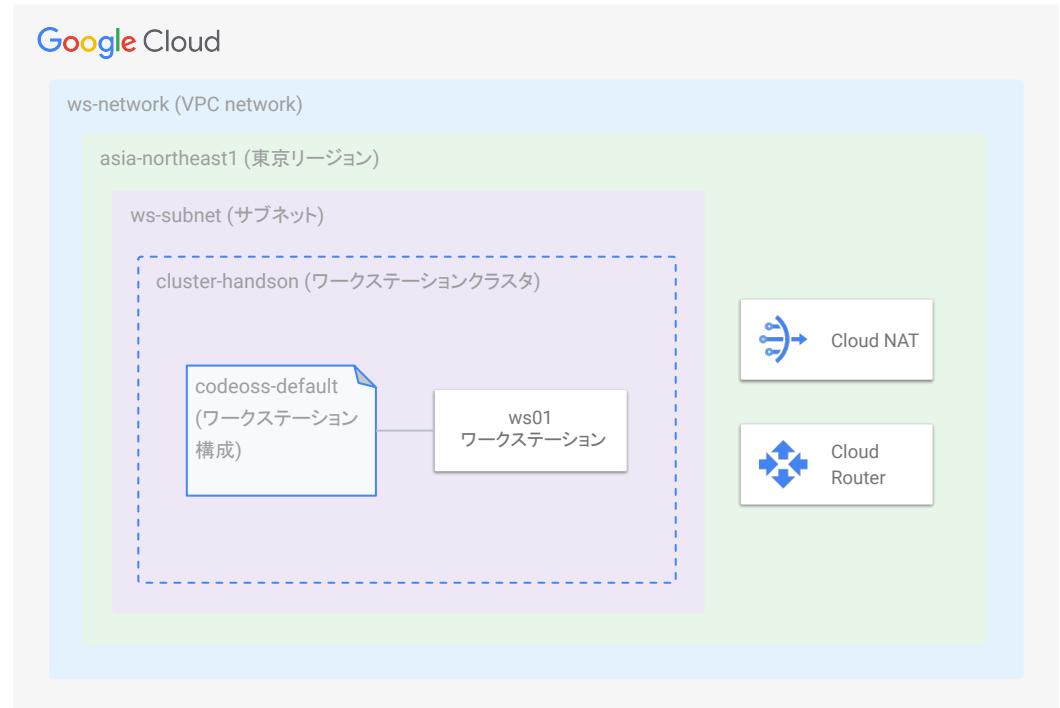
ステップ 4/19 ワークステーション用ネットワークの設定

1. VPC ネットワークの作成
2. サブネットの作成
3. Cloud Router の作成
4. Cloud NAT の作成
5. ワークステーションクラスタの作成

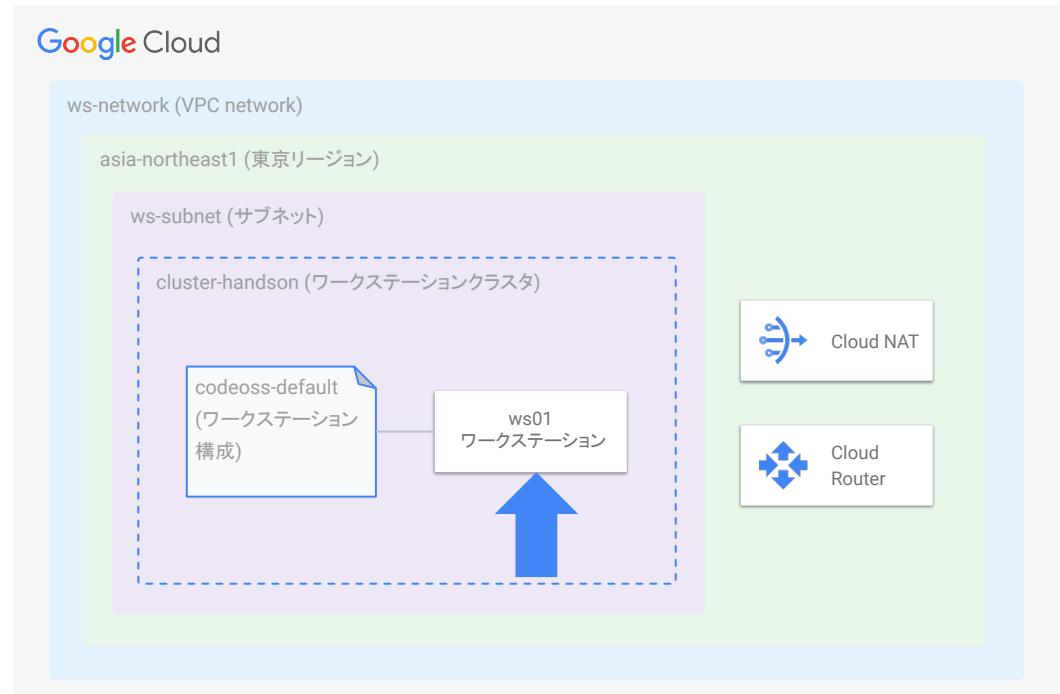


ステップ 6/19 基本のワークステーション構成から ワークステーションを作成する

1. ワークステーション構成の作成
2. ワークステーションの作成



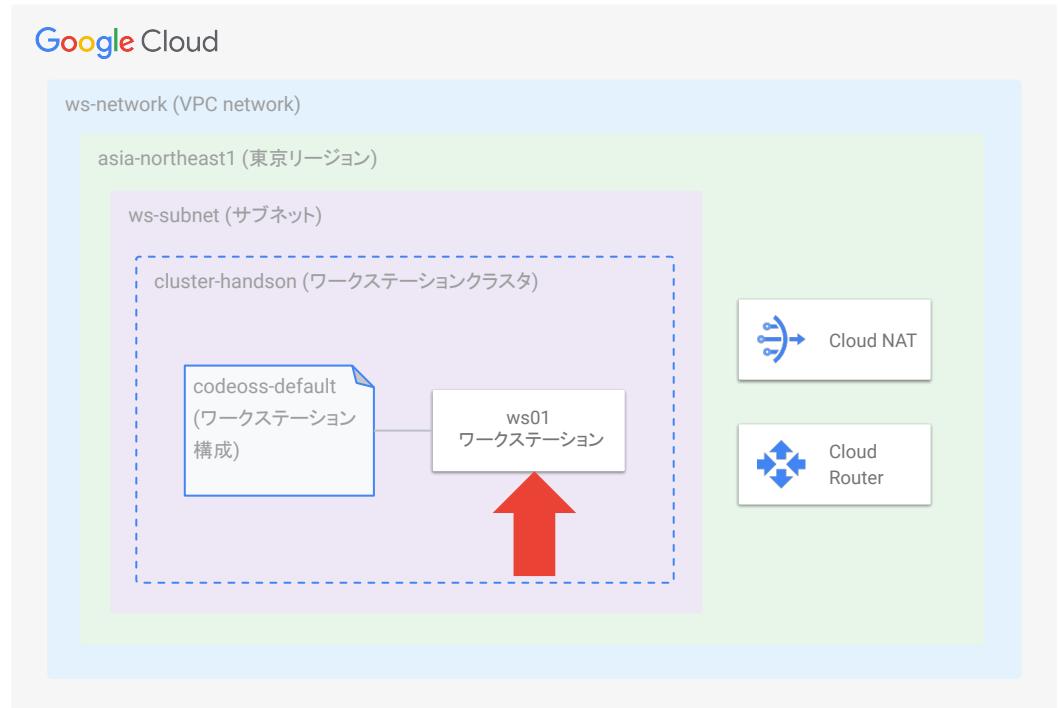
ステップ 7/19 開発者としてワークステーションを利用する



ステップ 8/19 管理者としてワークステーションを利用する

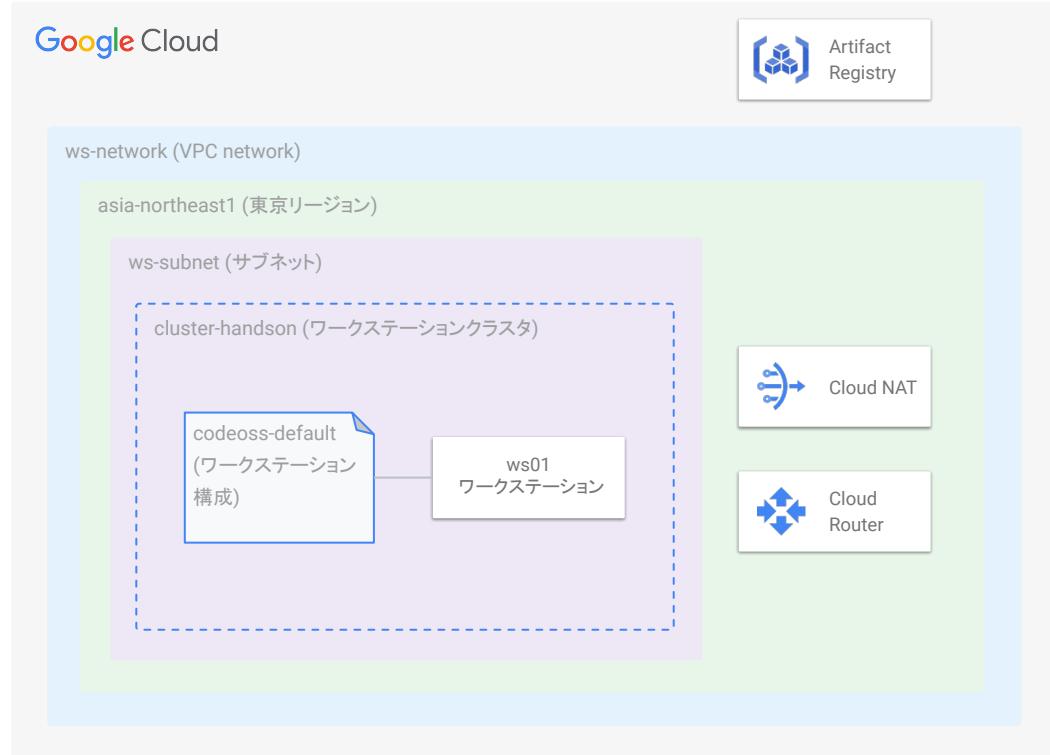
1. 稼働している

ワークステーションを停止する



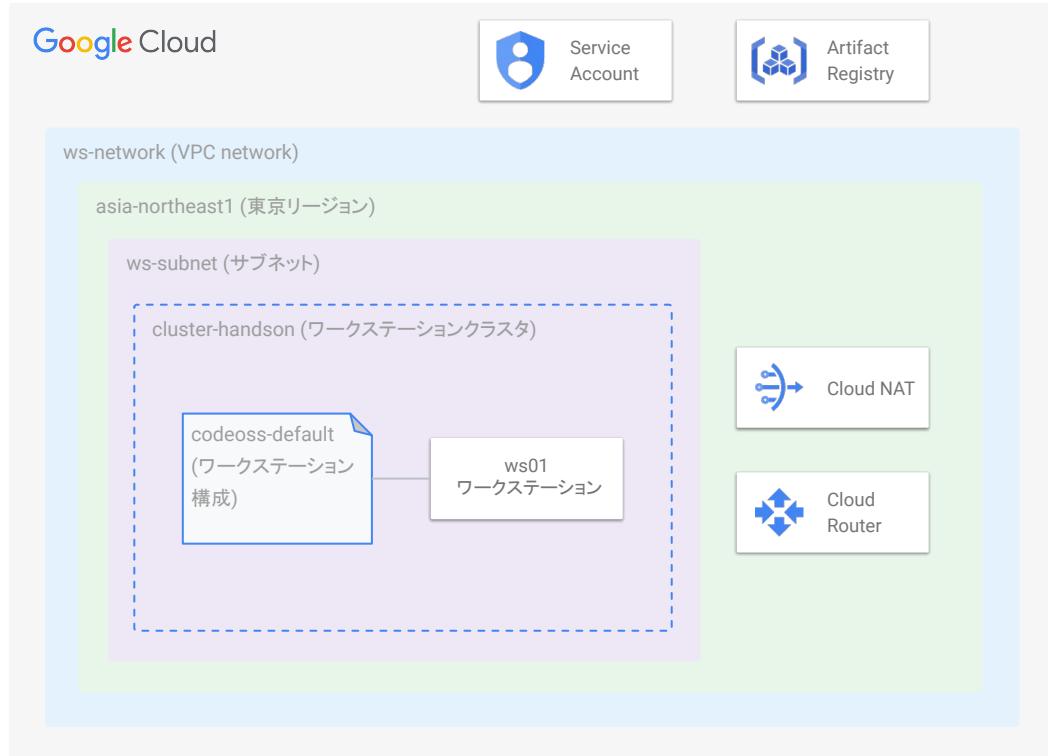
ステップ 9/19 コンテナイメージのカスタマイズ

1. Dockerfile の作成
2. Docker リポジトリ
(Artifact Registry) の作成
3. コンテナのビルド、プッシュ



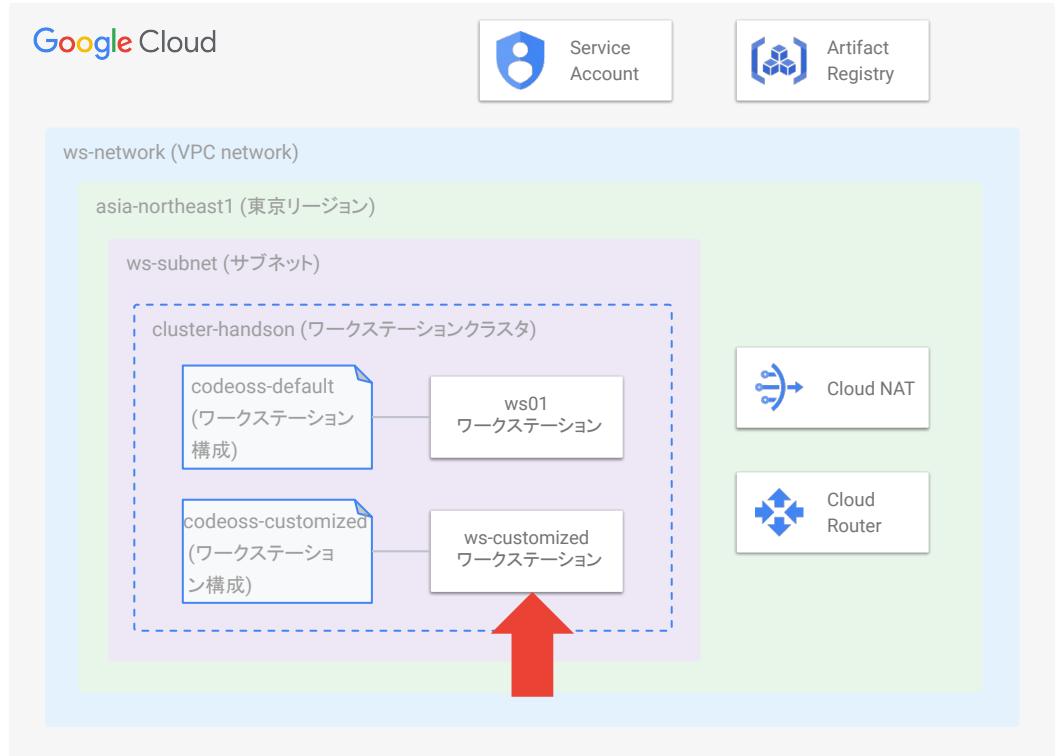
ステップ 10/19 サービスアカウントの設定

1. サービスアカウントの作成
2. サービスアカウントへの
Docker リポジトリ読み取り権限付与



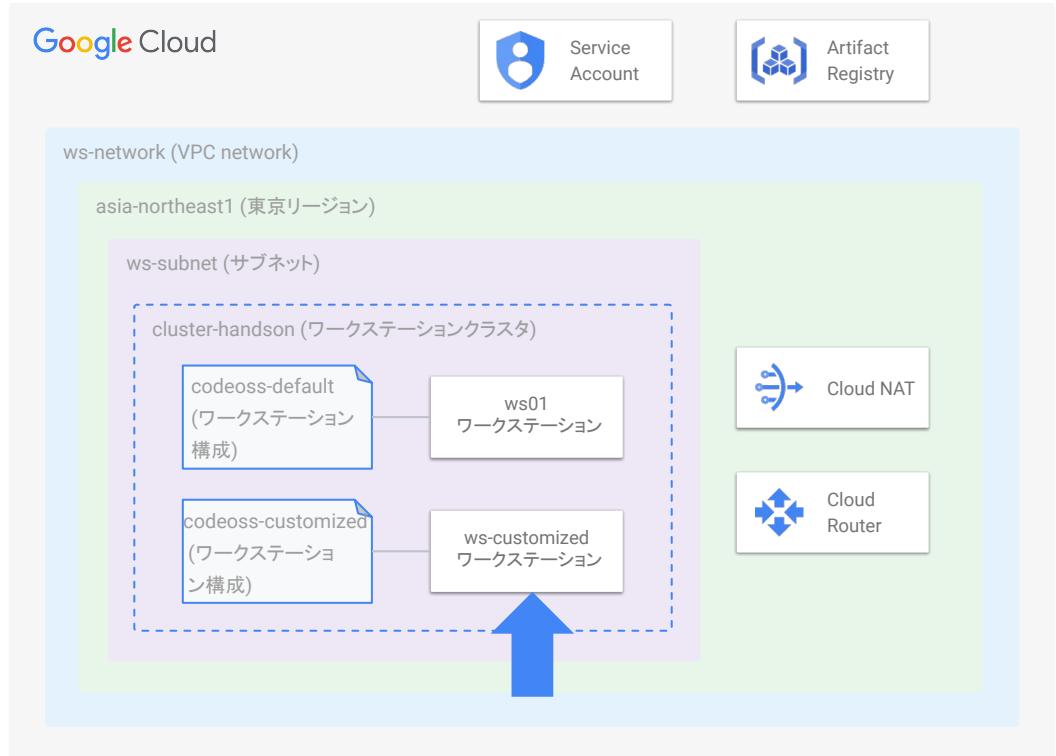
ステップ 11/19 カスタムイメージ (Dockerfile 利用) を使ったワークステーションの利用

1. サービスアカウントの作成
2. サービスアカウントへの Docker リポジトリ読み取り権限付与



ステップ 12/19 ホームディレクトリのカスタマイズ

1. カスタマイズスクリプトの作成
2. Dockerfile の更新
3. コンテナのビルド、プッシュ
4. ワークステーション構成の更新
5. カスタマイズの動作確認

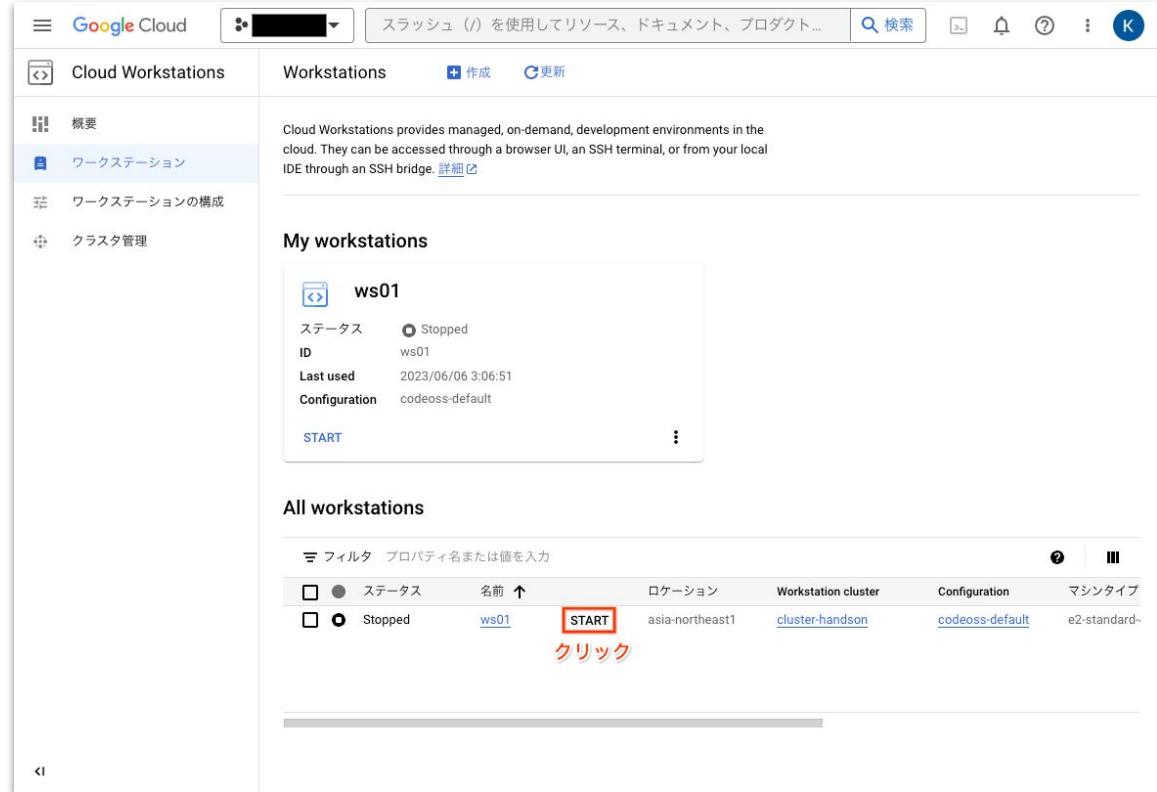


06

開発者として ワークステーションを利用する

1. ワークステーションを開始する

1. ワークステーション
一覧画面に遷移する
2. START をクリックする
3. 数分待つとステータスが
 Running に変化する

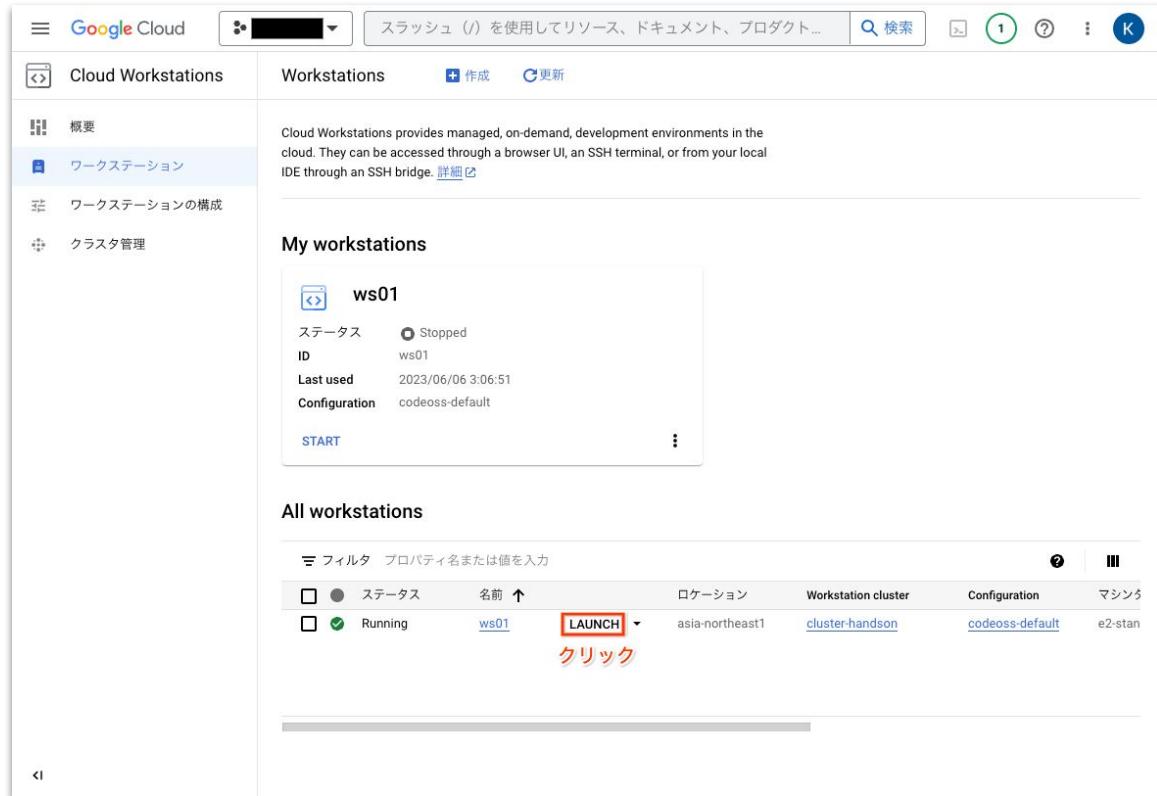


The screenshot shows the Google Cloud Platform interface for Cloud Workstations. The left sidebar has 'Cloud Workstations' selected under 'Workstations'. The main area displays 'My workstations' with one entry: 'ws01'. The details for 'ws01' show it is 'Stopped' (radio button selected), with ID 'ws01', last used on '2023/06/06 3:06:51', and configuration 'codeoss-default'. A 'START' button is visible. Below this, a table titled 'All workstations' lists 'ws01' again, showing it is still 'Stopped'. A red box highlights the 'START' button for 'ws01', with the text 'クリック' (Click) written below it.

フィルタ	名前	ロケーション	Workstation cluster	Configuration	マシンタイプ	
<input type="checkbox"/>	<input checked="" type="radio"/> Stopped	ws01	asia-northeast1	cluster-hands-on	codeoss-default	e2-standard-

2. ワークステーションにアクセスする

1. **LAUNCH** をクリックする
2. ブラウザから別のタブが開き
Code OSS for Cloud Workstations と表示される



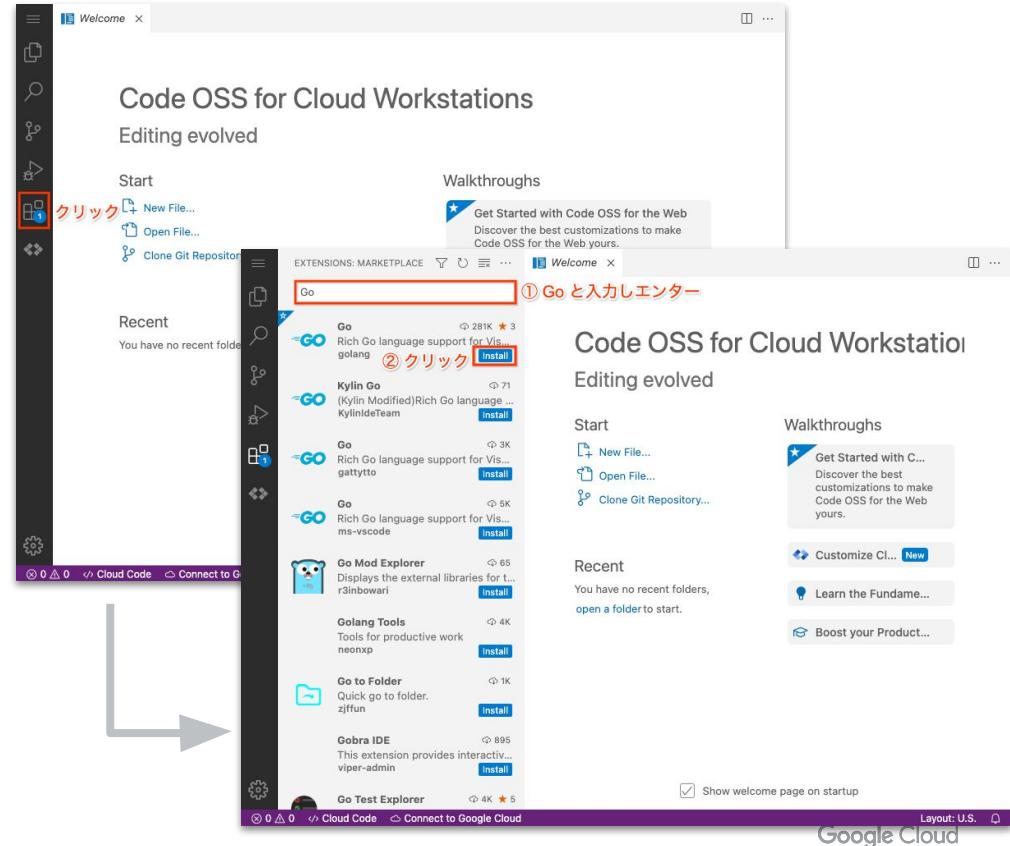
The screenshot shows the Google Cloud Cloud Workstations interface. The left sidebar has tabs for '概要' (Overview) and 'ワークステーション' (Workstations), with 'ワークステーション' currently selected. The main area displays information about a single workstation named 'ws01'. Below this, a section titled 'All workstations' lists one entry:

ステータス	名前	ロケーション	Workstation cluster	Configuration	マシン名
<input type="checkbox"/> Running	ws01	asia-northeast1	cluster-hands-on	codeoss-default	e2-stan

A red box highlights the 'LAUNCH' button next to the 'ws01' entry, with the word 'クリック' (click) written below it.

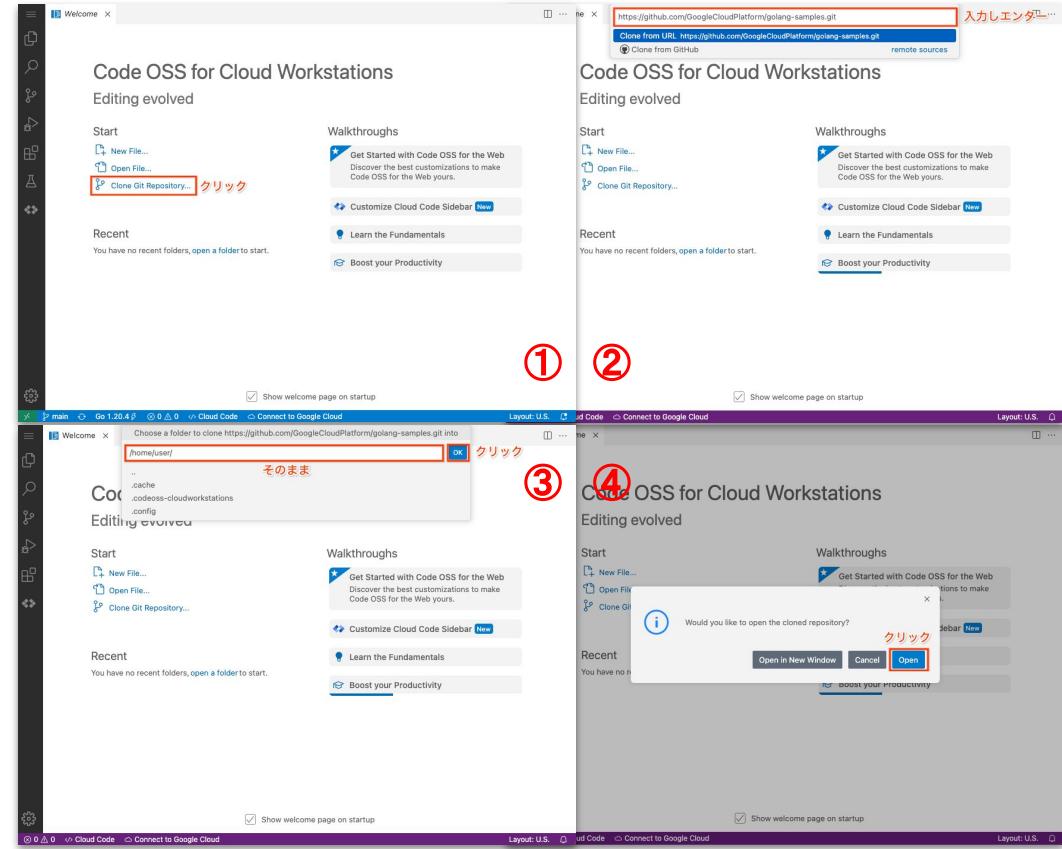
3. 拡張機能 (Extensions) をインストールする

1. 左メニューの中から下から 2 つめの **Extensions** をクリック
2. 検索フォームに **Go** と入力しエンター
3. 一番上にでてくる拡張機能の **Install** ボタンをクリック
4. 無事インストールされると手順 3 でクリックした **Install** ボタンが消える



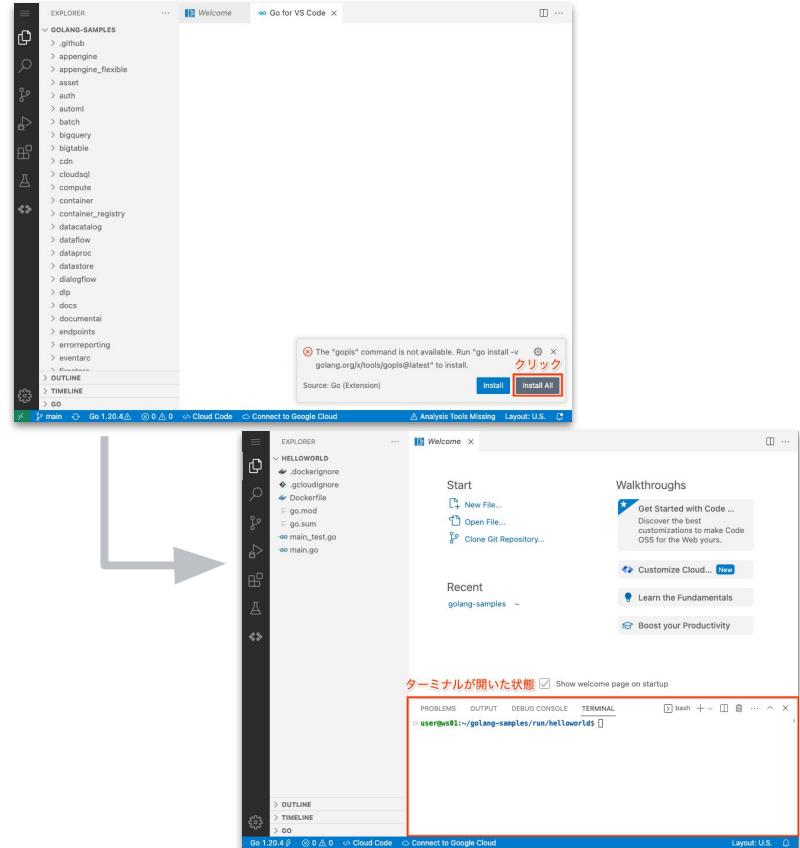
4. アプリケーションをクローンする

1. 再び LAUNCH から
ワークステーションにアクセス
2. **Clone Git Repository...** をクリック
3. Repository URL に以下を
コピー&ペーストし入力しエンター
 - <https://github.com/GoogleCloudPlatform/golang-samples.git>
4. 保存フォルダは **/home/user/** の
まで OK をクリック
5. Open ボタンをクリック



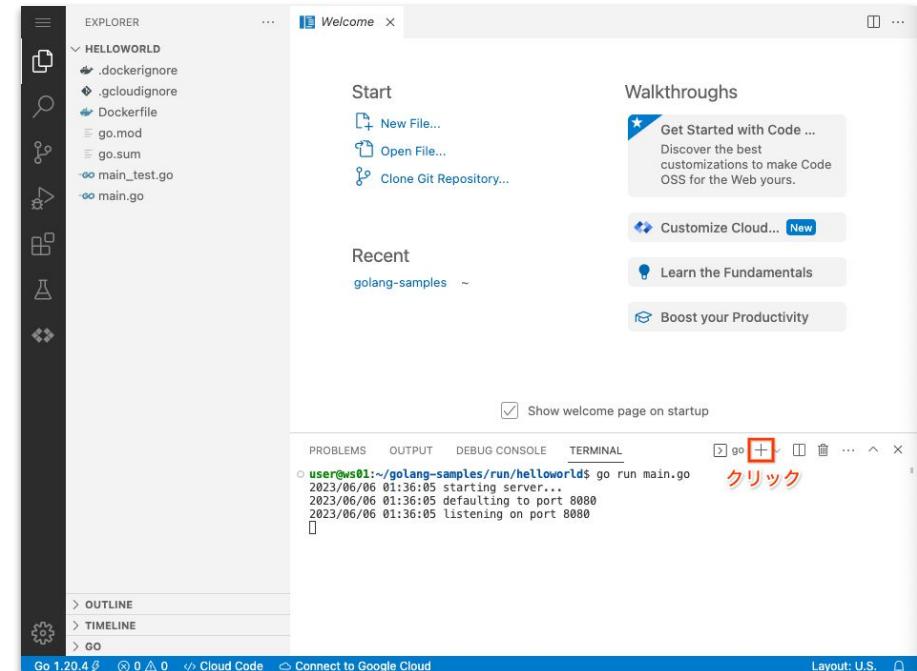
5. アプリケーションの実行準備

1. **Install All** をクリックし Go (Extension) の関連ツールをインストール
2. 左上のメニュー → File → Open Folder... から /home/user/golang-samples/run/helloworld/ を開く
※ このあとに Git (Extension) のダイアログが出た場合は Never をクリックする
3. 左上のメニュー → Terminal → New Terminal から ターミナルを開く



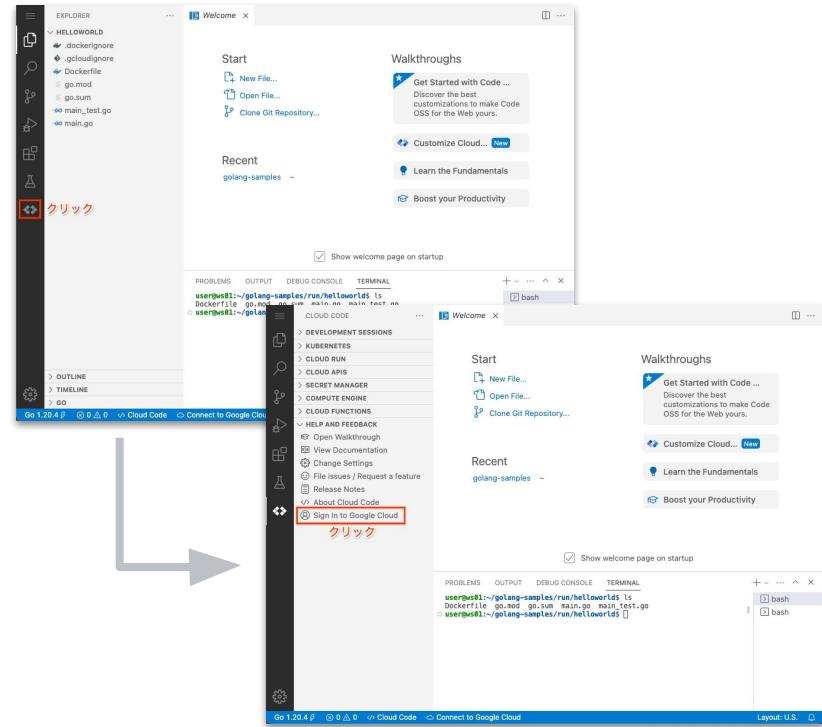
6. アプリケーションを実行する

1. ターミナルから以下を実行し
アプリケーションを起動する
`go run main.go`
2. + をクリックし新しいターミナルを開く
3. ターミナルから以下を実行し 8080 ポートの
URL を取得する
`echo http://localhost:8080`
4. 出力された URL をクリックしブラウザから
Helloworld が開くことを確認する
5. 元々のターミナルから `Ctrl-C` を入力し
アプリケーションを停止する



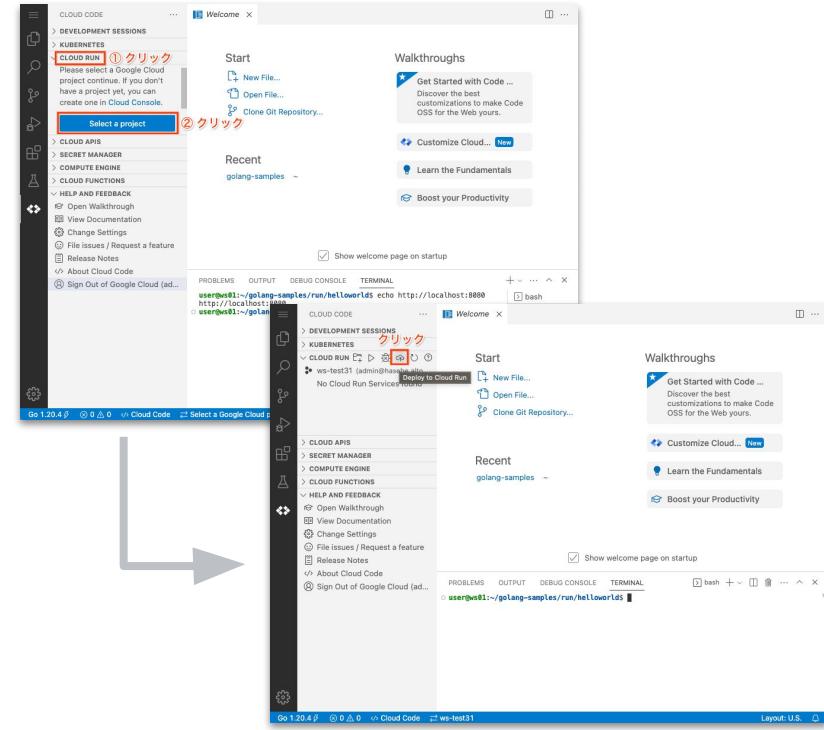
7. Cloud Code を使い、Google Cloud にサインインする

1. 左メニュー群最下部の **Cloud Code** をクリックする
2. Cloud Code 最下部の **Sign In to Google Cloud** 、**Proceed to sign in** の順にクリックする
3. ターミナルに表示された URL をクリック、確認画面から **Open** をクリックする
4. Google アカウントを選択、次に許可をクリックする
5. **Copy** ボタンをクリック、コピーした確認コードをターミナルから入力し、サインインできたことを確認する



8. Cloud Code を使い、Cloud Run にデプロイする

1. Cloud Code から CLOUD RUN 、Select a project の順にクリックし、利用しているプロジェクトを選択する
2. CLOUD RUN メニューの Deploy to Cloud Run をクリックする
* Deploy to Cloud Run ウィンドウに何も表示されない場合は、Deploy to Cloud Run ボタンの開き直し、ブラウザのリロードを試してみてください
3. 最下部の Deploy ボタンをクリック
4. デプロイ完了後、出力された URL をコピー、ブラウザから開いて動作確認



9. チュートリアルに戻る

GUI から ”開発者としてワークステーションを利用する” ステップはここで終了です。

チュートリアルが開いている Google Cloud コンソールに戻り、

”管理者としてワークステーションを利用する” を再開します。

Thank you.