AWS Black Belt Online Seminar

Amazon CodeCatalyst

Dev Environments 編

髙柴 元

Solutions Architect
2023/12

自己紹介

名前：髙柴 元(Gen TAKASHIBA)

所属：アマゾンウェブサービスジャパン合同会社
技術統括本部インダストリソリューション第二部
ソリューションアーキテクト

経歴：

情報通信・製造業の情報システム部、インフラエンジニアを経て2021年よりAWSにソリューションアーキテクトとして勤務

好きなAWSサービス：AWS Amplify, Amazon ECS

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、 AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

ご感想は X (Twitter) へ！ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では資料作成時点(2023/10)のサービス内容および価格についてご説明しています。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (https://aws.amazon.com/) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます
- 技術的な内容に関しましては、有料の AWS サポート窓口へお問い合わせください
- 料金面でのお問い合わせに関しましては、カスタマーサポート窓口へ
  お問い合わせください (マネジメントコンソールへのログインが必要です)

Black Belt Amazon CodeCatalyst シリーズ

- Overview 編
- Spaces 編
- Projects, Blueprints 編
- Source repositories 編
- **Dev Environments 編**
- Workflow 編
- Issues 編
- Identity, permissions, and access 編
- Extensions 編

シリーズ構成

- 全体像をお伝えする Overview 編
- 各機能の詳細についてお伝えする各機能編

シリーズの対象読者

- チーム開発をするすべてのアプリケーション開発者

Dev Environments 編の対象読者

- チーム開発で、開発環境の依存関係を揃えたい方
- ブランチごとに独立した開発環境を作りたい開発者
- クラウドで開発環境を持ちたい開発者

アジェンダ

1. Dev Environmentsの概要
2. Dev Environmentsの設定とライフサイクル
3. Devfileの定義
4. Dev EnvironmentsとIDEとの連携
5. Dev Environmentsの注意事項

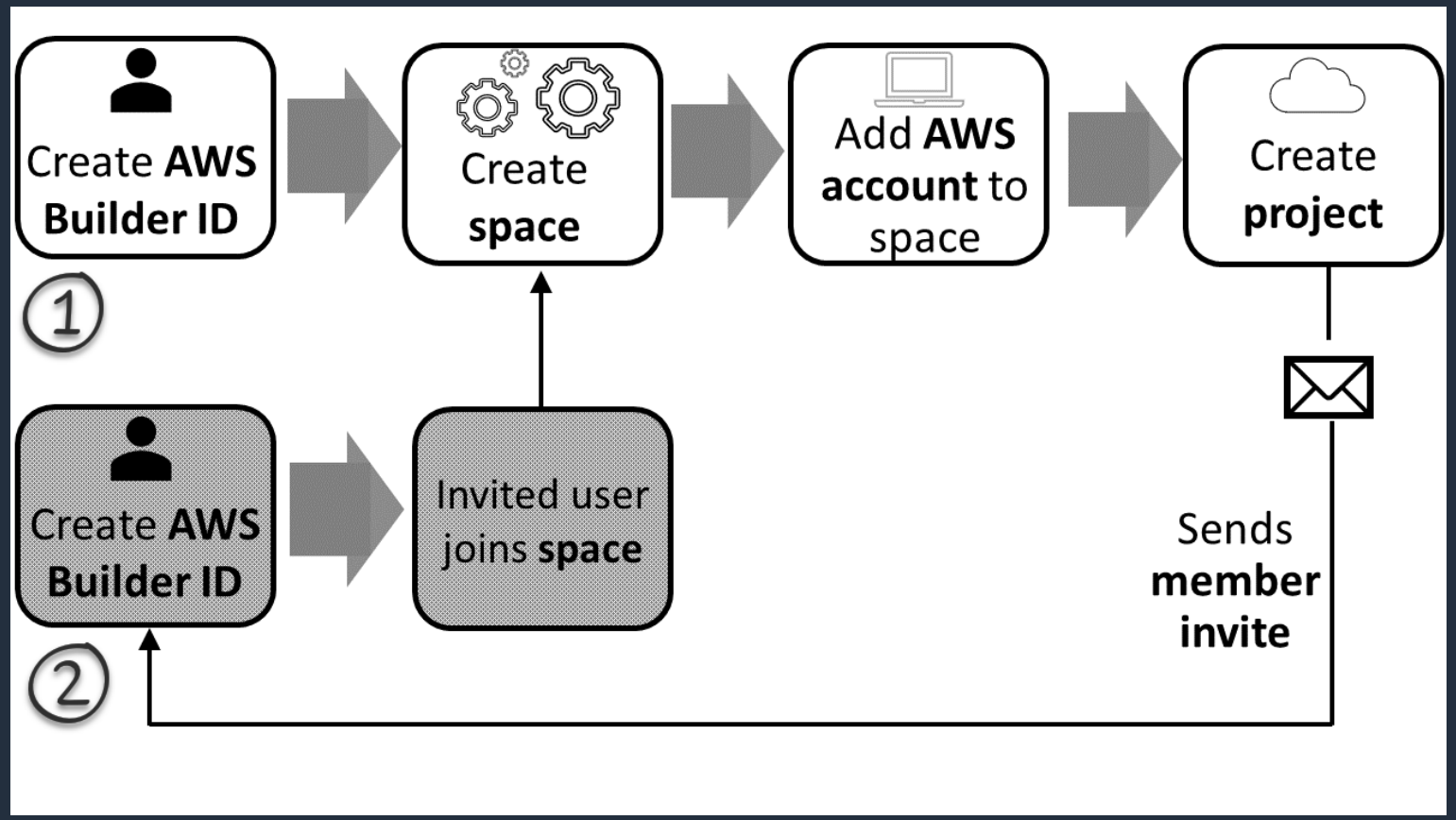
Dev Environmentsの概要

CodeCatalyst Dev Environments とは

- **オンデマンドで利用可能**なクラウドベースのLinux開発環境
- 依存関係・言語ランタイムなどの**必要なパッケージを Devfile を使い事前定義**しておき、プリインストールされた状態で起動可能
- Compute能力は **Free Tierで 2vCPU / 4GB RAM**、Standerd Tier で最大16vCPU / 32GB RAMまで拡張可能(ストレージは環境あたり16GB)
- Cloud9, **Visual Studio Code, IntelliJ IDEA, PyCharm, GoLand**から直接起動・接続が可能

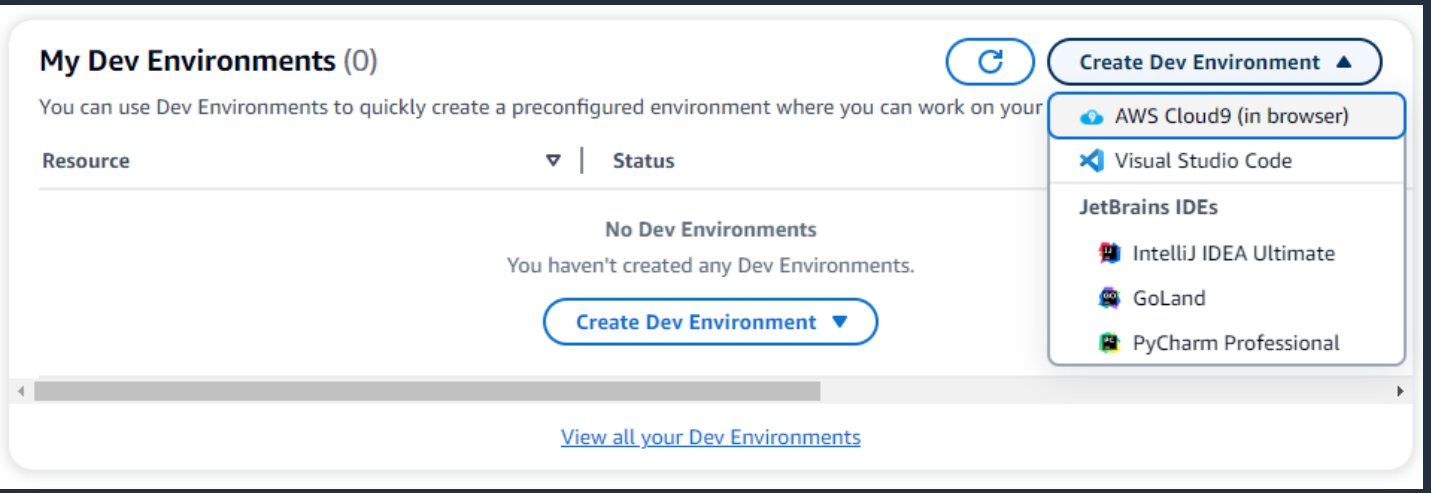
CodeCatalyst Dev Environments の開始方法①

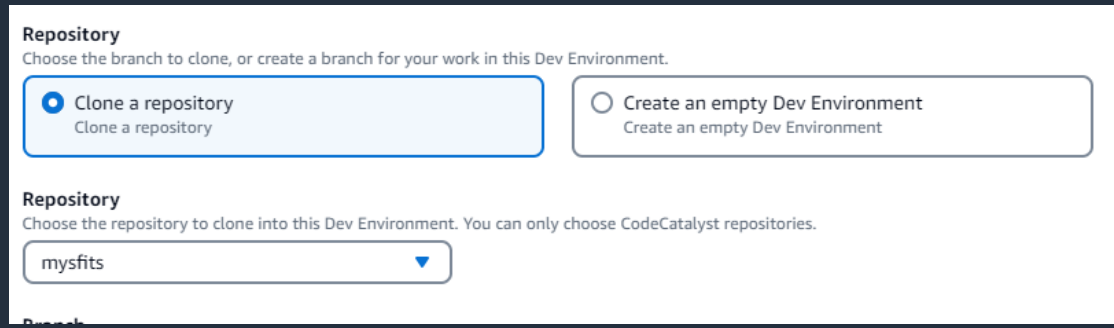
- まず、CodeCatalyst でプロジェクトを作成する



CodeCatalyst Dev Environments の開始方法②

- Create Dev Environment から利用するエディタを選択します
  - リポジトリをクローンして起動することもできますし、環境のみを起動することもできます

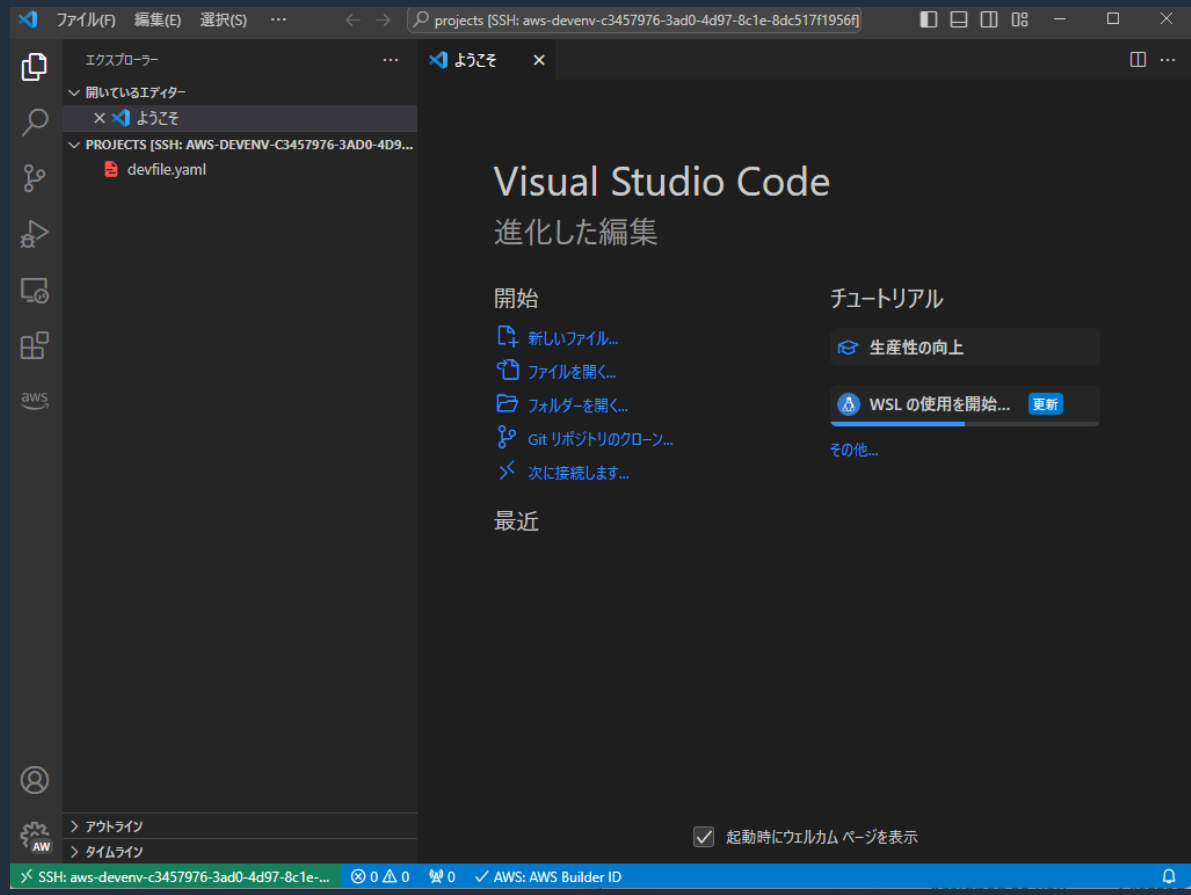




  - Create を押下後、選択したエディタが起動します
    - IDEとして Cloud9 を選択した場合、新しいタブで Cloud9 が開きます

CodeCatalyst Dev Environments の開始方法②

- 選択したエディタが SSH で Dev Environment に接続した状態で起動します
- Dev Environment(略称:DevEnv)はユーザー単位のリソースです。原則として他のユーザーが操作することはできません
- リポジトリを指定した場合、 /projects に clone された状態でスタートします
- リポジトリを含まないか、リポジトリに devfile が無かった場合は /projects 直下にデフォルト設定で devfile.yaml が作成されます

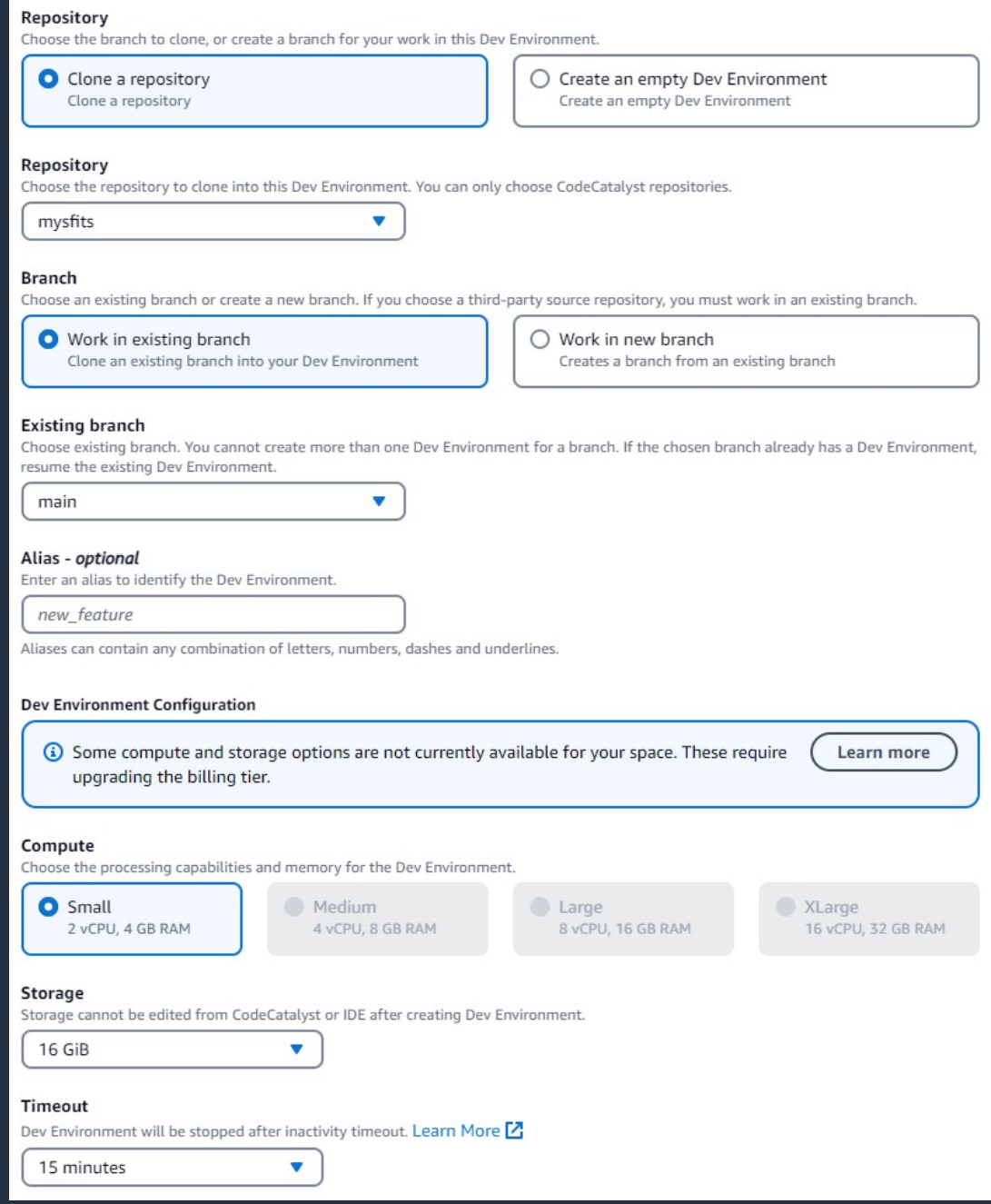


Dev Environments の設定とライフサイクル

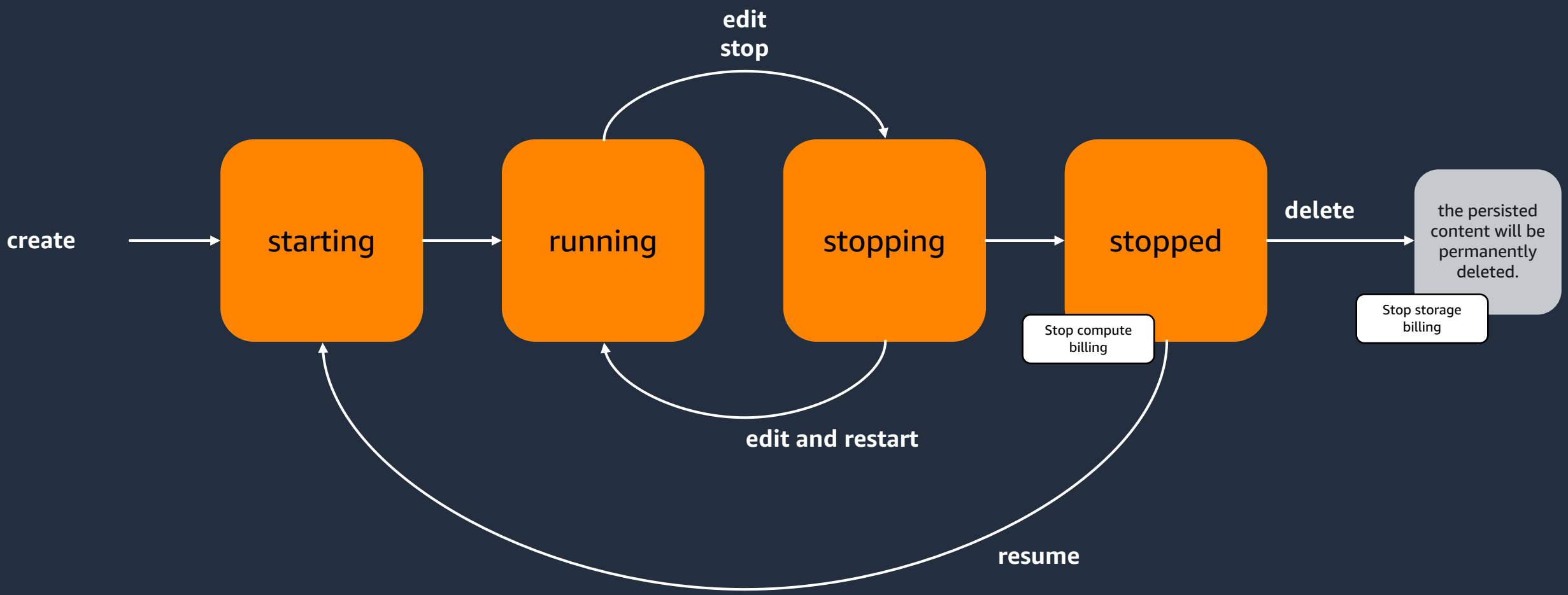
Dev Environments の設定項目

- Dev Environments そのものの設定
  - リポジトリ・ブランチ(optional)
  - エイリアス(optional)
  - vCPU, メモリ(~16vCPU, 32GB)
  - ストレージ(~64GB)
    - 作成後の変更不可
  - タイムアウト時間(~8hours)

- Dev Environments 内部の設定
  - リポジトリ内のDevfile
    - commands
    - events
    - components



Dev Environments のライフサイクル

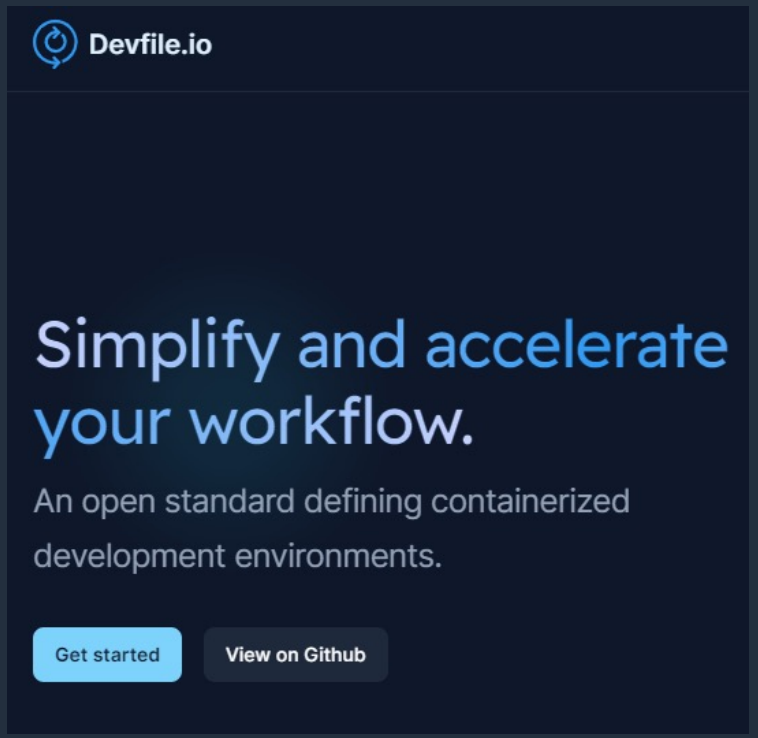


Devfile の定義

Devfile とは

- コンテナ化された開発環境を定義するオープンスタンダード
  - 実態は YAML ファイルで、 https://devfile.io/ にてスキーマが公開されています（CodeCatalyst は Ver.2 に対応）

- 必要なプロジェクトツールやアプリケーションライブラリを devfile に設定することで、それらを開発環境にインストールします



Devfile の全容

- 全要素のスキーマは devfile の WEB サイトで確認可能
  - https://devfile.io/docs/2.0.0/devfile-schema
- CodeCatalyst で利用できる devfileの機能は下記の通り
  - command
    - exec
  - event
    - postStart
  - component
    - args
    - env
    - mountSources
    - volumeMounts

```yaml
schemaVersion: 2.0.0
metadata:
  name: al2
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      mountSources: true
      command: ['sleep', 'infinity']
  - name: dockerstore
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "/projects/devfiles/script.sh"
  - id: yumupdate
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - setupscript
    - executescript
    - yumupdate
```

components

- CodeCatalystでは、devfile の仕様で定義されているtypeの中で、container のみ利用できます。指定したコンテナイメージが、ログインする DevEnv の環境となる
- コンテナイメージはパブリックレジストリを指定
- コンテナのエントリーコマンドが長期実行されない場合は、['sleep', 'infinity'] を command に指定して、コンテナが終了しないようする
- サンプルで指定している要素の他、args, env, volumeMounts を指定可能

```yaml
schemaVersion: 2.0.0
metadata:
  name: al2
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      mountSources: true
      command: ['sleep', 'infinity']
  - name: dockerstore
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "/projects/devfiles/script.sh"
  - id: yumupdate
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - setupscript
    - executescript
    - yumupdate
```

components 複数コンテナの同時起動

- container を複数定義することで、テストやモックに必要なローカルコンポーネントを DevEnv の内部であらかじめ起動することができます
- 右の例は、DynamoDB local のコンテナを起動してローカル環境で DynamoDB のテストを実行できるようにしています

https://aws.amazon.com/jp/blogs/news/managing-dev-environments-with-amazon-codecatalyst/

devfile.yaml

myenv: main / devfile.yaml

```yaml
schemaVersion: 2.0.0
metadata:
  name: aws-universal
  version: 1.0.1
  displayName: AWS Universal
  description: Stack with AWS Universal Tooling
  tags: ["aws"]
  projectType: "aws"
components:
  - name: aws-runtime
    container:
      image: public.ecr.aws/aws-mde/universal-image:latest
      mountSources: true
      volumeMounts:
        - name: docker-store
          path: /var/lib/docker
  - name: docker-store
    volume:
      size: 16Gi
  - name: dynamodb
    container:
      image: amazon/dynamodb-local
      mountSources: false
      endpoints:
        - name: dynamo
          targetPort: 8000
          exposure: public
```

commands

- DevEnv の内部で利用できるコマンドのショートカットを定義します。
  - DevEnv にログイン後 /aws/mde/mde command <command-id> で定義済みの処理を実行可能
  - command で定義した内容のみ、events プロパティ内でも利用可能
- idは全て小文字で定義

```yaml
schemaVersion: 2.0.0
metadata:
  name: al2
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      mountSources: true
      command: ['sleep', 'infinity']
  - name: dockerstore
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "/projects/devfiles/script.sh"
  - id: yumupdate
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - setupscript
    - executescript
    - yumupdate
```

events

- DevEnv のライフサイクルで実行可能なフック。現在は postStart のみ利用可能
- 実行する commands の id を配列で渡して利用

```yaml
schemaVersion: 2.0.0
metadata:
  name: al2
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      mountSources: true
      command: ['sleep', 'infinity']
  - name: dockerstore
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "/projects/devfiles/script.sh"
  - id: yumupdate
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - setupscript
    - executescript
    - yumupdate
```

Universal devfile imageについて

- CodeCatalyst が用意している各言語とツールセットを含んだデフォルトイメージ。現在 version 3.0 まで公開済み
  - 2023/12時点では latest タグが1.0を指すことに注意
- 新規イメージがリリースされた際に発行される SNS トピックが公開されている
  - arn:aws:sns:us-east-1:089793673375:universal-image-updates をサブスクライブして任意のSNSサブスクリプションを作成可能
- 各イメージに含まれている言語バージョンはドキュメントを参照
  - https://docs.aws.amazon.com/codecatalyst/latest/userguide/devenvironment-universal-image.html

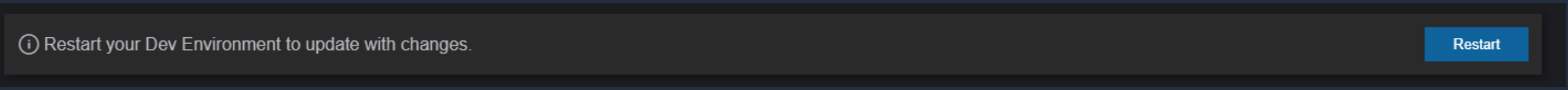
```
components:
  - name: aws-runtime
    container:
      image: public.ecr.aws/aws-mde/universal-image:1.0
      mountSources: true
      volumeMounts:
        - name: docker-store
          path: /var/lib/docker
  - name: docker-store
    volume:
      size: 16Gi
```

devfileの編集・反映方法

- devfile の変更を反映するためには、変更後の devfile で新しく DevEnv を作成するか、CLI でツール実行が必要です
  1. devfile を変更・保存する
  2. $ /aws/mde/mde status で変更有無を確認する
     - devfile の変更が検知されている場合、status が CHANGED になっています

```
[mde-user@ip-10-130-46-148 projects]$ /aws/mde/mde status
{
    "status": "CHANGED",
    "location": "myenv/devfile.yaml"
}
```

  3. $ /aws/mde/mde start --location {devfileの場所}
     - AWS Toolkit 上の restart ボタンでも同じコマンドが実行されます
     - 実行後、DevEnv のコンテナが再起動します。一時的に接続が切断されることがありますが、自動で再接続されます。



devfile 編集のデバッグ

- status 確認
  - $ /aws/mde/mde status の結果に location が含まれない場合、実行時にエラーが発生してリカバリーモードに入っている
    - https://docs.aws.amazon.com/codecatalyst/latest/userguide/devenvironment-devfile.html#devenvironment-devfile-recovery
- 各種ログ
  - /aws/mde/logs/devfile.log
    - devfile 自体の実行ログ（start --locationのログはここ）
  - /aws/mde/logs/devfileCommands.log
    - devfile の commands 実行ログ

ログサンプル

devfile.log

devfile.log

```
1  2023-12-06T01:26:01.787Z INF Local Docker images found: [{"Repository":"public.ecr
2  2023-12-06T01:26:01.787Z INF Starting devfile
3  2023-12-06T01:26:05.017Z INF Executing post start devfile commands
4  2023-12-06T01:33:41.64Z INF Local Docker images found: [{"Repository":"public.ecr.
5  2023-12-06T01:33:41.64Z INF Starting devfile
6  2023-12-06T01:33:56.201Z INF Executing post start devfile commands
7  2023-12-06T01:52:24.233Z INF Local Docker images found: [{"Repository":"public.ecr
8  2023-12-06T01:52:24.233Z INF Starting devfile
9  2023-12-06T01:52:24.353Z ERR Failed to start devfile: failed to run command: [202
10 2023-12-06T01:52:24.368Z ERR failed to run command: [2023-12-06 01:52:24] failed t
11 2023-12-06T01:52:24.454Z ERR Failed to start provided devfile: failed to run comma
12 2023-12-06T01:53:01.671Z INF Local Docker images found: [{"Repository":"public.ecr
13 2023-12-06T01:53:01.671Z INF Starting devfile
14 2023-12-06T01:53:15.2Z INF Executing post start devfile commands
15
```

devfileCommands.log

postStart に sudo yum -y update --security を設定したサンプル

devfileCommand

```
114 --> Finished Dependency Resolution
115
116 Dependencies Resolved
117
118 ================================================================================
119  Package                        Arch   Version                Repository  Size
120 ================================================================================
121 Installing:
122  kernel-devel                   x86_64 4.14.330-250.540.amzn2 amzn2-core  13 M
123 Updating:
124  avahi-libs                     x86_64 0.6.31-20.amzn2.0.3    amzn2-core  61 k
125  ctags                          x86_64 5.8-23.amzn2           amzn2-core 156 k
126  gawk                           x86_64 4.0.2-4.amzn2.1.3      amzn2-core 873 k
127  gmp                            x86_64 1:6.0.0-15.amzn2.0.3   amzn2-core 276 k
128  indent                         x86_64 2.2.11-13.amzn2.0.3    amzn2-core 149 k
129  java-17-amazon-corretto-devel  x86_64 1:17.0.9+8-1.amzn2.1   amzn2-core 146 k
130  kernel-headers                 x86_64 4.14.330-250.540.amzn2 amzn2-core 1.2 M
131  openssl                        x86_64 1:1.0.2k-24.amzn2.0.11 amzn2-core 497 k
132  openssl-devel                  x86_64 1:1.0.2k-24.amzn2.0.11 amzn2-core 1.5 M
133  openssl-libs                   x86_64 1:1.0.2k-24.amzn2.0.11 amzn2-core 1.2 M
134  openssl11-libs                 x86_64 1:1.1.1g-12.amzn2.0.19 amzn2-core 1.4 M
135  ruby                           x86_64 2.0.0.648-36.amzn2.0.6 amzn2-core  74 k
136  ruby-irb                       noarch 2.0.0.648-36.amzn2.0.6 amzn2-core  95 k
137  ruby-libs                      x86_64 2.0.0.648-36.amzn2.0.6 amzn2-core 2.8 M
138  rubygem-bigdecimal             x86_64 1.2.0-36.amzn2.0.6     amzn2-core  85 k
139  rubygem-io-console             x86_64 0.4.2-36.amzn2.0.6     amzn2-core  57 k
140  rubygem-json                   x86_64 1.7.7-36.amzn2.0.6     amzn2-core  82 k
141  rubygem-psych                  x86_64 2.0.0-36.amzn2.0.6     amzn2-core  85 k
142  rubygem-rdoc                   noarch 4.0.0-36.amzn2.0.6     amzn2-core 325 k
143  rubygems                       noarch 2.0.14.1-36.amzn2.0.6  amzn2-core 217 k
144  vim-common                     x86_64 2:9.0.2120-1.amzn2.0.1 amzn2-core 8.0 M
145  vim-data                       noarch 2:9.0.2120-1.amzn2.0.1 amzn2-core  83 k
146  vim-enhanced                   x86_64 2:9.0.2120-1.amzn2.0.1 amzn2-core 1.6 M
147  vim-filesystem                 noarch 2:9.0.2120-1.amzn2.0.1 amzn2-core  77 k
148  vim-minimal                    x86_64 2:9.0.2120-1.amzn2.0.1 amzn2-core 755 k
149  xxd                            x86_64 2:9.0.2120-1.amzn2.0.1 amzn2-core  94 k
```

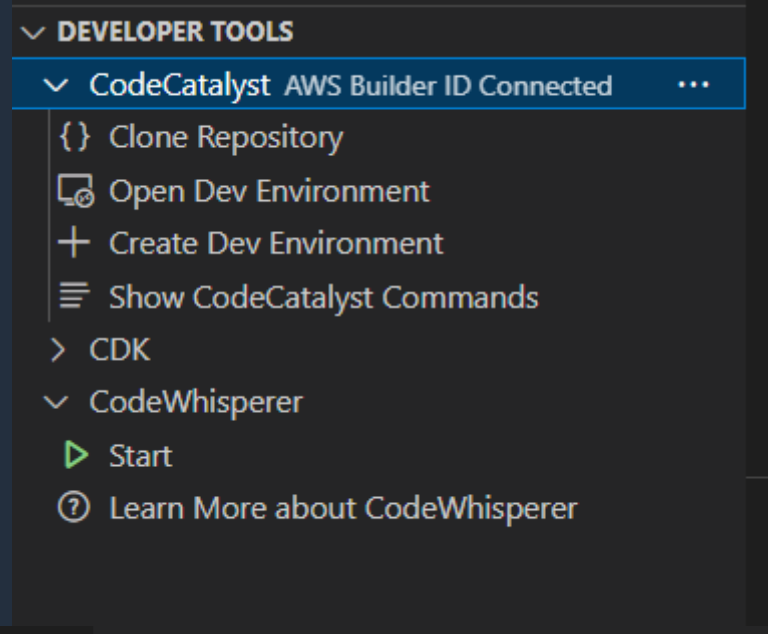
Dev EnvironmentsとIDEの連携

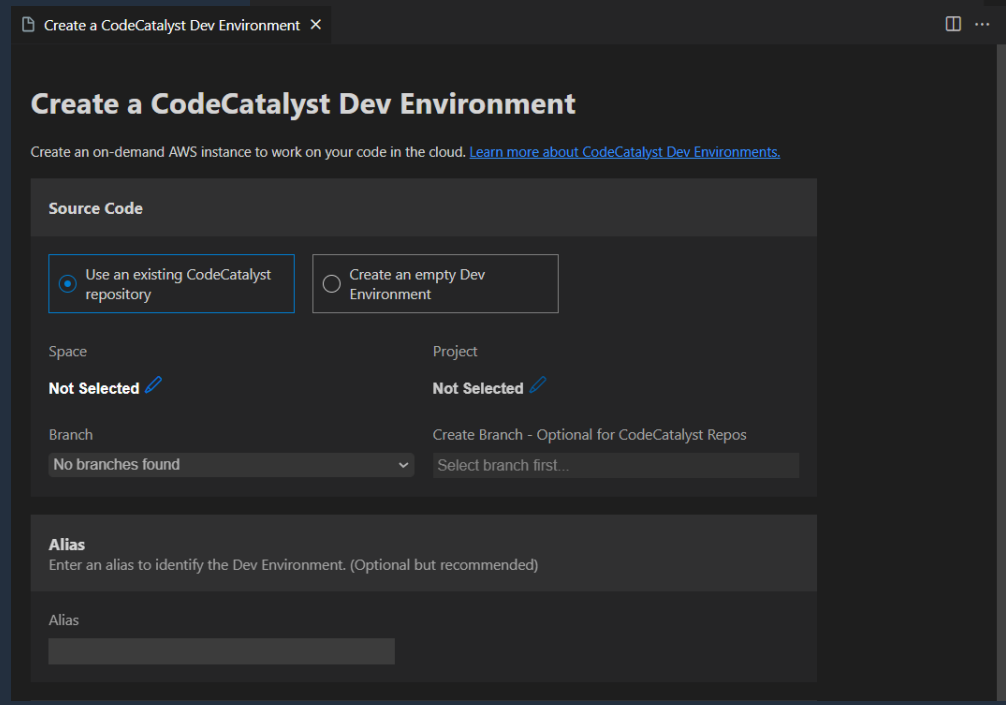
AWS Toolkitによる連携

- AWS Toolkit に AWS Builder ID で接続すると、CodeCatalyst メニューが利用可能

- できること
  - リポジトリの Clone
  - Dev Environment の接続と作成
  - devfileもしくはDevEnv設定変更後の再起動

- AWS Builder ID による Dev Environment や CodeWhisperer の利用と IAM 認証情報による AWS リソース表示を併用可能





Dev Environmentsの料金

追加料金なしで利用可能なリソース ※2023 年 12 月 時点

カテゴリ	Free tier	Standard tier	Enterprise tier
Dev Environment 稼働時間	60 時間	200 時間	160 時間 x ユーザー数
Dev Environment 合計ストレージ容量	64 GB	128 GB	64 GB x ユーザー数
データ転送 (アウト)	10 GB	10 GB	10 GB

https://codecatalyst.aws/explore/pricing

Tier と利用可能な機能 ※2023 年 12 月 時点

一部 Tier でのみ利用可能な機能や選択可能なリソースが存在する

機能・リソース	Free tier	Standard tier	Enterprise tier
Dev Environment あたりの ストレージ容量	16 GB	16 GB, 32 GB, 64 GB	16 GB, 32 GB, 64 GB
利用可能なインスタンスタイプ (Linux)	2 vCPU / 4 GB	2 vCPU / 4 GB 4 vCPU / 8 GB 8 vCPU / 16 GB 16 vCPU / 32 GB	2 vCPU / 4 GB 4 vCPU / 8 GB 8 vCPU / 16 GB 16 vCPU / 32 GB

https://codecatalyst.aws/explore/pricing

* Linux のみ

追加リソース料金 ※2023 年 12 月 時点

Standard tier　Enterprise tier

- Dev Environment 稼働時間

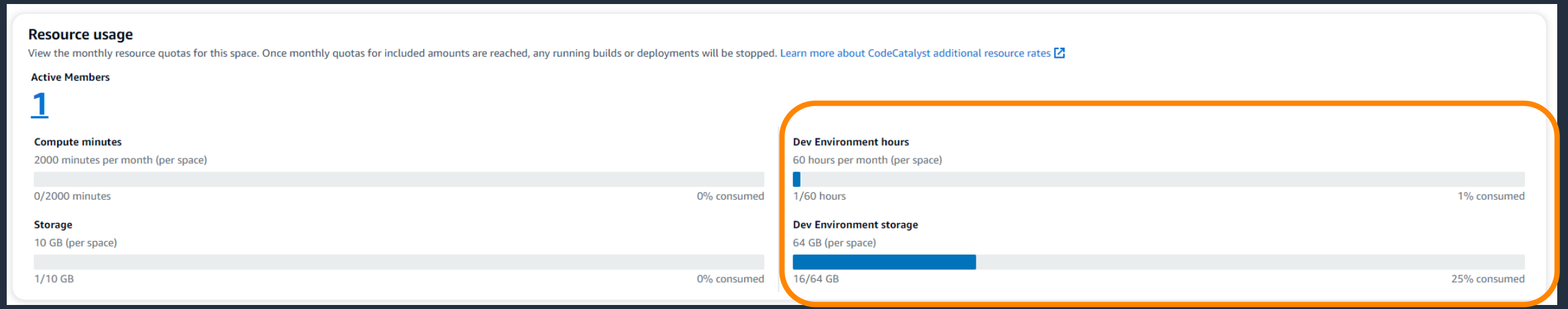
OS	インスタンスタイプ	オンデマンド料金	プロビジョンド料金
Linux	2 vCPU / 4 GB	-	$0.12 / 時間
	4 vCPU / 8 GB	-	$0.24 / 時間
	8 vCPU / 16 GB	-	$0.46 / 時間
	16 vCPU / 32 GB	-	$0.92 / 時間

- Dev Environment 合計ストレージ容量: $0.05 / GB-月
- データ転送 (アウト): EC2 データ転送料金に準ずる

https://codecatalyst.aws/explore/pricing

利用量の確認方法

Space の Setting > Billing から Space 内の利用量を確認できる



Thank you!