

Dockerで手に入れるデプロイ環境

2018-06-30 [kamijin-fanta](#)

<https://kinoko-hoge.connpass.com/event/88048/>

自己紹介

- @kamijin-fanta
- インフラな会社
- Scala, TypeScript

目標

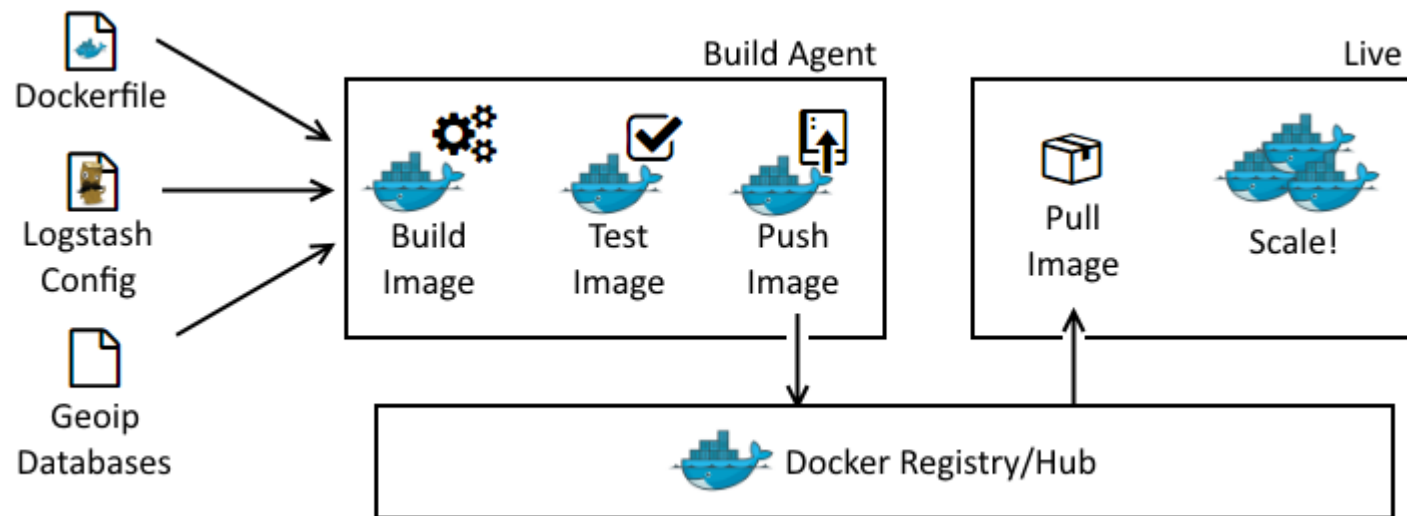
- 周辺技術の名前・関係性
- k8s構築方法を知る
- CIを回して、モダン・安全に本番アプリケーションをデプロイする

※注

- エディタをDockerの中に閉じ込めるとかマニアックな話じゃないです
- ざっくり理解するための資料なので、細部まで正確性を求めないでください

何故Dockerでデプロイ環境を作るか

- なぜDocker/k8s
 - 高可用性
 - インフラ・アプリケーションの分離→可搬性
- 何故Dockerで開発環境を作るか
 - 環境を統一させやすい
 - ミドルウェア・アプリケーション・ライブラリ / 開発・・・ステージング・本番



コンテナとオーケストラレータ

Webアプリケーションデプロイ史

- apache/tomcatなどのサーバをセットアップ・ftp/rsyncでアップロード→個人端末のビルド環境に依存
- リモートサーバーにVCSで同期・手動でビルドさせてデプロイ→個人端末・開発/本番でバージョン管理難しい
 - ex: サーバ・言語バージョンのアップデートでコードが変わる場合...
- 仮想マシンのイメージをビルドして展開→リソースオーバーヘッドが大きい・オートスケールが間に合わない
- 初期PaaS(Google App Engine/初期のheroku)→言語バージョンなどがプラットフォームに縛られる
- コンテナ→手元・開発・本番で各種バージョンを合わせるのが容易で、リソース消費が比較的少ない

可用性を上げたい

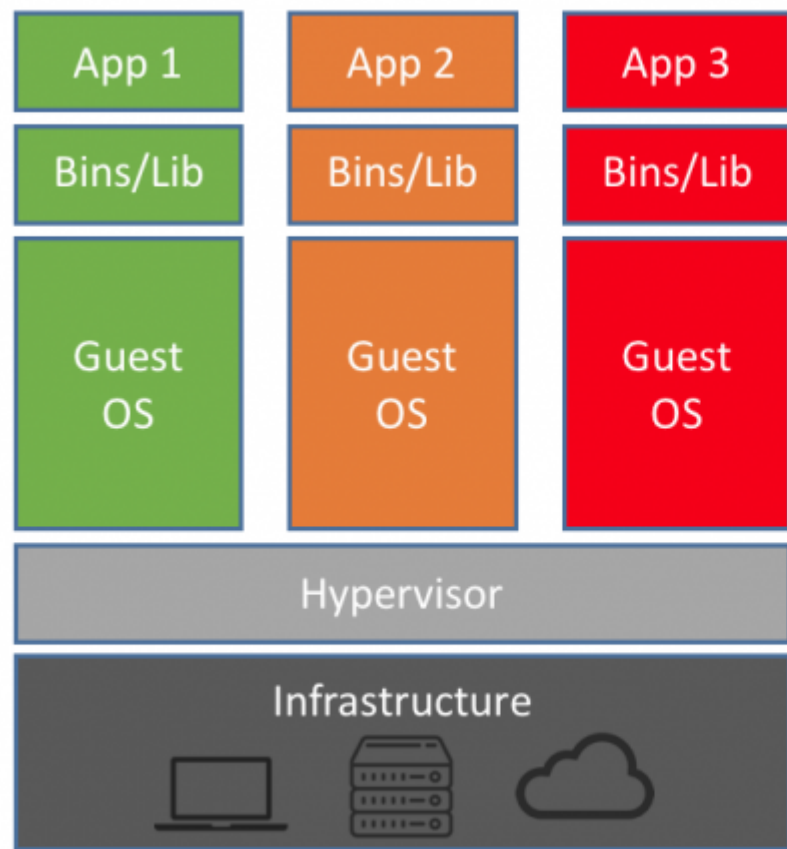
- 1台構成 → 落ちたらしんどい
- アクティブスタンバイ → 片方が完全に余剰リソースになる
- クラスタリング → 複数のマシンを束ねて一つのリソースとして見る・数台壊れた程度でサービスに影響が出ないように設計・リソース効率改善

ソリューション

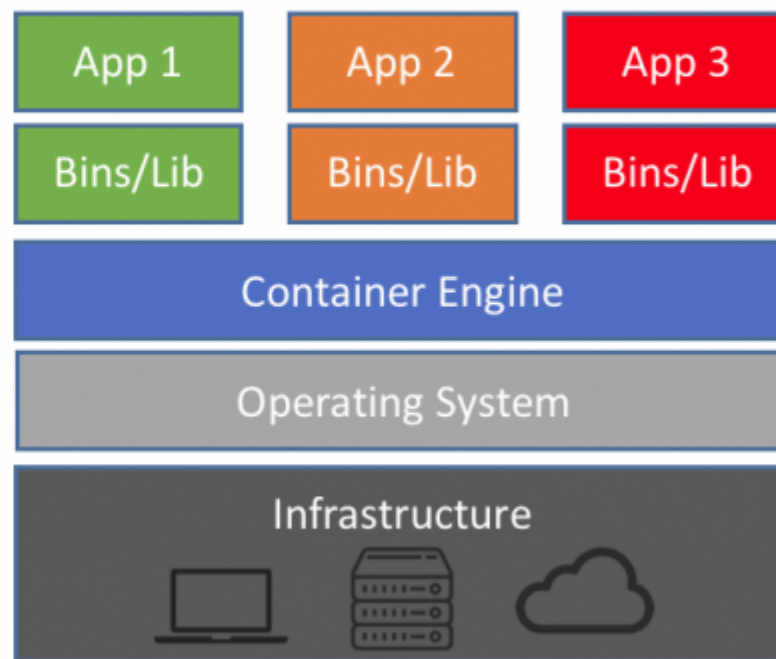
- コンテナ
 - アプリケーションをイメージにまとめる
 - 仮想マシンより軽量
 - どこでも同じように動かす事ができるようにする
 - 手元/dev/stage/prodでの環境統一ができる
 - デプロイ改善 (属人性排除)
- コンテナオーケストラレーション
 - コンテナをいい感じに指定した数どこかで起動してくれる
 - 必要なメモリ容量・ディスク容量などを指定して空いているノードを割り当ててくれる
 - 落ちたら別のところで再起動
 - 可用性向上
 - リソース効率改善

コンテナ技術

- lxd
 - 対応する技術はKVM
 - 複数のアプリケーションをまとめてイメージに焼く
 - 標準的な構成をコンテナ化して、環境のコピーなどを用意にする
 - [Java, Redis, MySQL]
- Docker
 - 1つのコンテナに1つのアプリケーションを配置する
 - サンドボックス
 - 複数コンテナを組み合わせる
 - [Java] [Redis] [MySQL]



Machine Virtualization



Containers

<https://blog.netapp.com/blogs/containers-vs-vms/>

オーケストラレーション技術

- DC/OS (mesos/marathon/metronome)
 - MesosでSpark/Cassandraなどを使っていると、リソースの共有が可
 - 定期的なジョブ実行(Cron)
 - コンテナを指定した数どこかで実行する
- kubernetes (略: k8s)
 - yamlでインフラを記述
 - コミュニティ標準になりつつ有る
 - 複数のアプリケーションを **サービス** 単位でデプロイ
- swarm
 - Docker標準/今後k8s互換に？
 - DockerCompose

コンテナオーケストレーションの細かい動作

- クラスタリング
- ネットワーキング
 - VXLAN/UDP/ipip/bgp
 - gcp/aws
- 監視
 - HTTPポート監視・コマンド実行
 - コンテナ再起動
- ログ管理
- ローリングアップグレード

チュートリアル

- 実際に、どのように開発環境・本番環境を作っていくかを見る
 - ローカルでのテスト・ビルド
 - CIでのテスト・ビルド
 - クラスタでCI・デプロイ

ローカルでのdocker build

- 1つのDockerImageを配布する単純な例

<https://github.com/kamijin-fanta/docker-example-2018>

- Reactのプロジェクト
 - nodeのバージョンは9で動かしたい
 - 依存ライブラリは `yarn` コマンドで取得
 - テストは `CI=true yarn test --ci` で実行
 - ビルドは `yarn build` で `build` ディレクトリに生成
- サーバはNginxを使いたい

Dockerfile

```
FROM node:9.11 AS build
```

```
WORKDIR /app
```

```
COPY package.json yarn.lock ./
```

```
RUN yarn
```

```
COPY . ./
```

```
RUN CI=true yarn test --ci
```

```
RUN yarn build
```

```
FROM nginx:1.15 AS web
```

```
COPY --from=build /app/build /usr/share/nginx/html
```

```
docker build -t docker-example-2018:0.0.1 .
```

```
docker run --name docker-example-2018 -d -p 8080:80 docker-example-2018:0.0.1
```

雑に1台のマシンにデプロイ

[Unit]

Description=docker-example-2018

Requires=docker.service

[Service]

Type=simple

ExecStart=/usr/bin/docker run --name docker-example-2018 -p 8080:80 docker-example-2018:

Restart=always

[Install]

WantedBy=multi-user.target

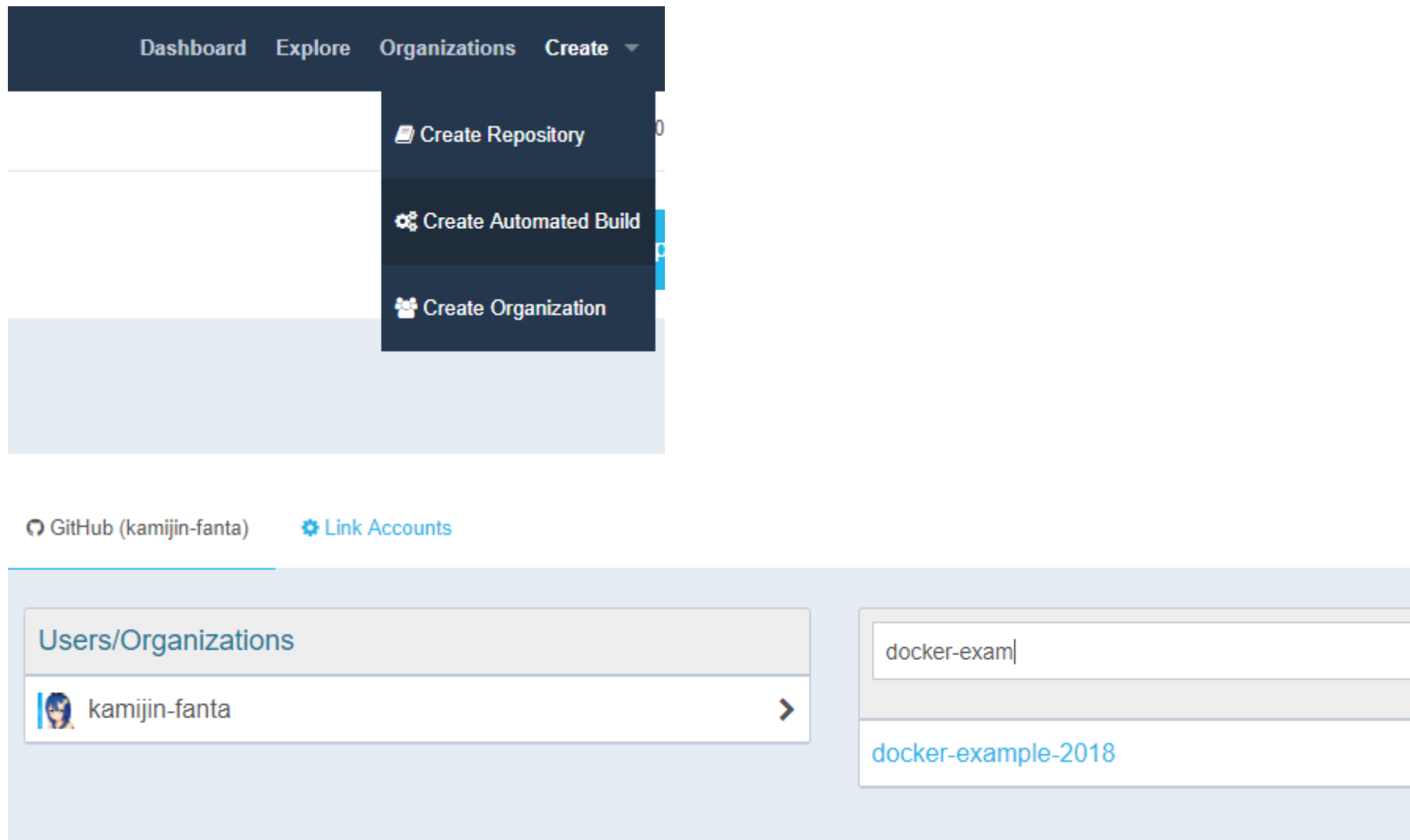
- ↑を `/etc/systemd/system/docker-example-2018.service` みたいな位置に設置する
- `systemctl enable docker-example-2018.service`
- `systemctl start docker-example-2018.service`

スケールさせる必要がない・個人開発ならこのくらいでも良い

CIを設定する

- さっきのGithubに置いたReactプロジェクトをDocker HubでCIする
- 公開リポジトリ・レジストリなら無料で出来る→OSS向け
- Webhookを設定すればPush時に自動的にビルドしてくれる

Docker hub 設定



リポジトリ選べる

PUBLIC | AUTOMATED BUILD

kamijin/docker-example-2018 ☆

Last pushed: 9 minutes ago

[Repo Info](#) [Tags](#) [Dockerfile](#) [Build Details](#) [Build Settings](#) [Collaborators](#) [Webhooks](#) [Settings](#)

Status	Actions	Tag	Created	Last Updated
✓ Success		latest	16 minutes ago	9 minutes ago

PUBLIC | AUTOMATED BUILD


kamijin/docker-example-2018 ☆

Last pushed: never

[Repo Info](#) [Tags](#) [Dockerfile](#) [Build Details](#) [Build Settings](#) [Collaborators](#) [Webhooks](#) [Settings](#)

Build Settings

☒ When active, builds will happen automatically on pushes.
The build rules below specify how to build your source into Docker images. The name can be a string or a regex. The Docker Tag name may contain variables. We currently support {sourceref}, which refers to the source branch/tag name. [Show more](#)

 Source Repository
[kamijin-fanta/docker-example-2018](#)

Type	Name	Dockerfile Location	Docker Tag Name		
Branch ▾	master	/	latest	+	Trigger
Branch ▾	All branches except master	/	Same as branch	-	

Save Changes

<https://hub.docker.com/r/kamijin/docker-example-2018/>

(割と時間掛かる...)

Google Container Builder

- 機能的にはDocker Hub+αという感じ
 - 基本非公開のレジストリにPushできる
- ビルド時間課金+ストレージ使用量の課金
 - プライベートなプロジェクトを多数ビルドするならこっちが良いかもしれない
- 割と早い

DockerHubとやることはあまり変わらないので、省略

<https://cloud.google.com/container-builder/docs/creating-build-triggers?hl=ja>

クラスタでCI・デプロイ

- コンテナのビルド・ビルド後のデプロイ先にDocker/k8sを使用
- k8sは検証目的なのでminikube(後述)で建てる
- k8sの上でjenkinsのCI環境を整える
- CIが完了すれば、自動的にアプリケーションがデプロイされるようにする
- 開発環境・本番環境を分離する

ci: 継続的インテグレーション (continuous integration)

様々なkubernetes利用形態

- 利用方法
 - マネージド
 - GCP GCE
 - AWS EKS
 - オンプレ
 - GCP, AWS, Azure
 - その他IaaS・ベアメタル
- 構築ツール
 - tectonic, rancher, minikube, etc...

jenkins-x

- Github/k8s環境に適したCIパッケージ
- ローカルのマシンからCLIで各種操作行える
- デプロイ構成をGithubのPRで管理する
 - どのアプリ・バージョンがデプロイされているか
- 今回はjenkins-xとデプロイ先を同じk8sに配置



Linuxマシンを用意

- GCPでも仮想マシンでもなんでもいいです
- 例では、さくらのクラウドに4G/4コア/ubuntu16を構築
- RAMをケチるとクラスタが崩壊するので注意

minikube

- 基本的にはDockerのインストールと、`minikube` コマンドの導入のみ
- Docker hubではなく、独自のDockerRegistryを建てるので、その設定を行う

```
# configure insecure-registry
echo '{ "insecure-registries":["10.0.0.0/8"] }' > /etc/docker/daemon.json
service docker restart

# start minikube
minikube start
```

- <https://github.com/kubernetes/minikube> の `Linux Continuous Integration without VM Support` を参考にしてください
- Docker導入: <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

Github設定

Settings / Developer settings

OAuth Apps
GitHub Apps
Personal access tokens

Edit personal access token

If you've lost or forgotten this token, you can regenerate it, but be aware that any scripts or applications using this token will need to be updated. [Regenerate token](#)

Token description

jenkins-x

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations

- GithubでPersonal Access Tokenを作成
- jxのインストールに複数回必要になるので控えておく
- 必要な権限は repo, user:email

jx install

- jxコマンドをダウンロード・ `jx install` でウィザードが開始
- GithubのKeyを作成: repo, user:email

```
# jx
curl -L https://github.com/jenkins-x/jx/releases/download/v1.2.140/jx-linux-amd64.tar.gz
sudo mv jx /usr/local/bin

# helm
curl -L https://storage.googleapis.com/kubernetes-helm/helm-v2.9.1-linux-amd64.tar.gz |
sudo mv linux-amd64/helm /usr/local/bin/

# dependency
apt install make socat

# install
jx install
```

詳細: <https://jenkins-x.io/getting-started/install-on-cluster/>

jxトラブルシューティング

- 一発でインストールできないと、色々ゴミが残ってしんどい
- インストールし直す前に以下のコマンドを打ってクリーンアップしてからやり直すことをオススメ

```
# k8sクラスタ上のjxのデプロイ・設定などを削除
kubectl delete ns jx
# 手元マシンに残っているjxへの接続情報・Githubの認証情報などを削除
rm -rf .jx
# k8sクラスタ上のhelmの削除
kubectl -n "kube-system" delete deployment tiller-deploy`
```

jxインストール完了

- コンソールに認証情報などが表示されるので控える

```
Jenkins X deployments ready in namespace jx
```

```
*****
```

```
NOTE: Your admin password is: *****
```

```
*****
```

```
Getting Jenkins API Token
```

```
unable to automatically find API token with chromedp using URL http://jenkins.jx.153.127
Please go to http://jenkins.jx.153.127.201.69.nip.io/me/configure and click Show API Tok
Then COPY the token and enter in into the form below:
```

勝手にできたエンドポイント・UI

- 様々なUIやAPIが追加された
- URLを確認するには `jx open`

```
# jx open
Name                                URL
jenkins                            http://jenkins.jx.153.127.201.69.nip.io
jenkins-x-chartmuseum              http://chartmuseum.jx.153.127.201.69.nip.io
jenkins-x-docker-registry           http://docker-registry.jx.153.127.201.69.nip.io
jenkins-x-monocular-api             http://monocular.jx.153.127.201.69.nip.io
jenkins-x-monocular-ui              http://monocular.jx.153.127.201.69.nip.io
nexus                              http://nexus.jx.153.127.201.69.nip.io
```

Jenkins

 **Jenkins**

6

検索

admin | ログアウト

Jenkins

自動リロードをon

新規ジョブ作成

開発者

ビルド履歴

プロジェクト相関関係

ファイル指紋チェック

Jenkinsの管理

Support

My Views

Open Blue Ocean

認証情報

New View

説明を記入

すべて +

S	W	名前 ↓	最新の成功ビルド	最新の失敗ビルド	ビルド所要時間	Fav
		kamijin-fanta	—	—	—	

アイコン: [S](#) [M](#) [L](#)

[凡例](#) [RSS 全ビルド](#) [RSS 失敗ビルド](#) [RSS 最新ビルドのみ](#)

Jenkins

Pipelines







Administration

Logout

Pipelines

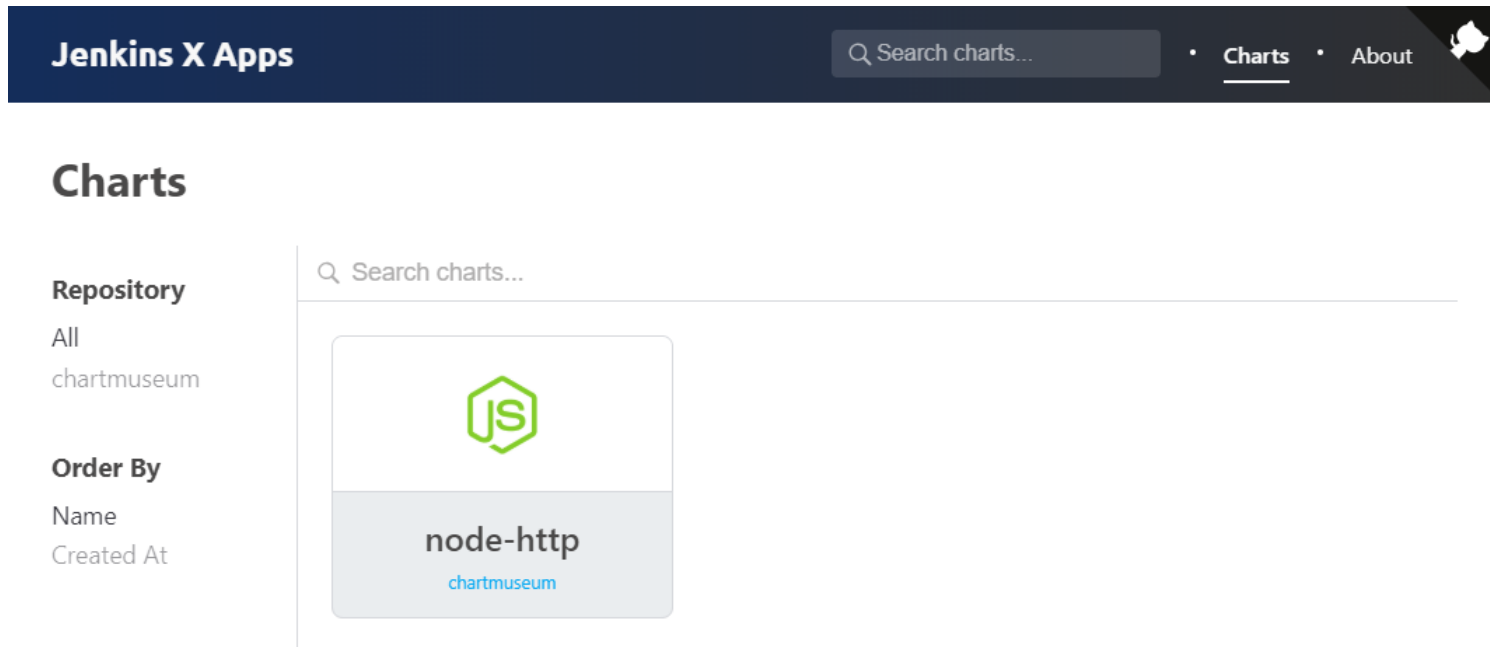
Search pipelines...

New Pipeline

NAME	HEALTH	BRANCHES	PR	
kamijin-fanta / environment-dukefield-production		1 passing	-	
kamijin-fanta / environment-dukefield-staging		1 passing	1 failing	
kamijin-fanta / node-http		1 passing	-	

Monocular

- アプリケーションのカatalogみたいなもの
- これからプロジェクトを作っていくが、ここに登録される
- 利用可能なアプリケーション・バージョンなどが見渡せる



Nexus

- Javaのアプリケーションのリポジトリ
- 今回は使わない

The screenshot shows the Nexus Repository Manager OSS 3.8.0-02 interface. The top navigation bar includes the Nexus logo, version information, a search bar, and links for refresh, help, and sign in. The left sidebar has a 'Browse' section with 'Welcome', 'Search', and 'Browse' options. The main content area features a 'Welcome' message, three notification banners (yellow, green, and purple), and two primary sections: 'Get Started' and 'Survey: Developer Tasks'. The 'Get Started' section lists 'Installation', 'Upgrading', 'Configuration', and 'Repository Formats'. The 'Survey: Developer Tasks' section contains a text input field for user feedback and a question about the user's primary role.

Nexus Repository Manager
OSS 3.8.0-02

Search components

Sign in

Welcome Welcome to Nexus Repository Manager!

Nexus Repository 3.12.1 is now available with an urgent fix to the Maven Upload UI (and the associated REST endpoint). [Release notes](#) [Download it now](#)

Upgrade now to add support for your repository Firewall. [Learn more](#)

Participate in the new Sonatype Community, come to learn from and share with your peers. [Check it out](#)

Get Started

- Installation**
[Get up and running](#)
- Upgrading**
[Upgrade to the latest version](#)
- Configuration**
[Set things up properly](#)
- Repository Formats**
Pre-installed Community supported

Survey: Developer Tasks

What did you do the last time you visited Nexus Repository Manager? Were you successful in accomplishing your task?

Your primary role related to your use of Nexus

Githubにリポジトリが勝手にできる

- staging/productionのデプロイ設定が記述されている

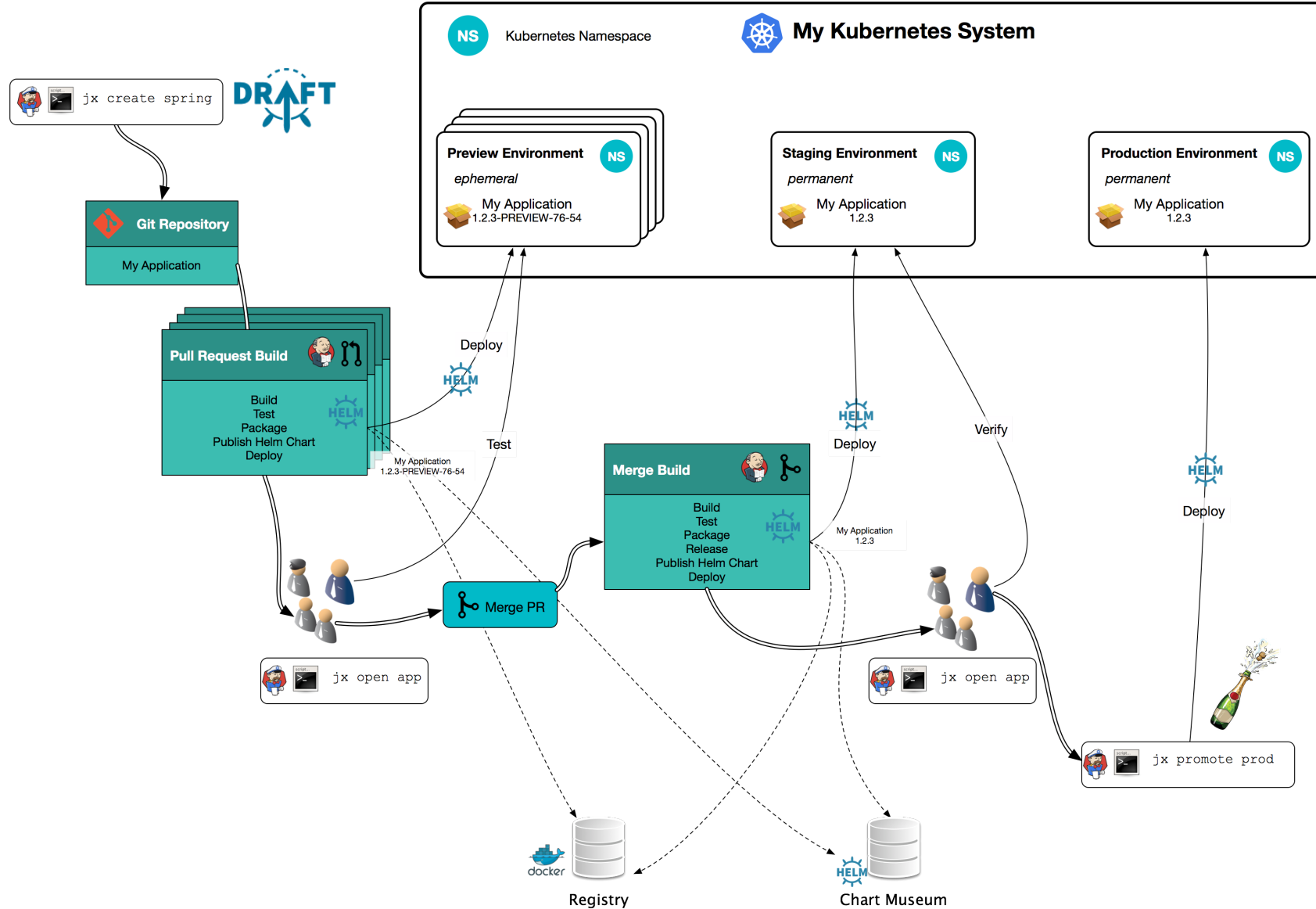
[environment-dukefield-production](#)

● Makefile 📄 Apache-2.0 Updated 29 seconds ago

[environment-dukefield-staging](#)

● Makefile 📄 Apache-2.0 Updated 43 seconds ago

構築されたデプロイフロー



クイックスタートでプロジェクトを作る

- ウィザード形式ですすめていく

```
root@minikube:~# jx create quickstart -f http
? select the quickstart you wish to create [Use arrows to move, type to filter]
  golang-http
> node-http
  python-http
  rust-http
  scala-akka-http-quickstart
  spring-boot-http-gradle
```

2018年6月現在 プロジェクト名・Githubのリポジトリ名を異なるものにすると、ビルドできなくなるバグが有るので注意

プロジェクト作成完了

Created Jenkins Project: <http://jenkins.jx.153.127.201.69.nip.io/job/kamijin-fanta/job/n>


Watch pipeline activity via: `jx get activity -f node-http-jenkins -w`
Browse the pipeline log via: `jx get build logs kamijin-fanta/node-http-jenkins/master`
Open the Jenkins console via `jx console`
You can list the pipelines via: `jx get pipelines`
When the pipeline is complete: `jx get applications`

For more help on available commands see: <http://jenkins-x.io/developing/browsing/>

Note that your first pipeline may take a few minutes to start while the necessary docker

Creating github webhook for kamijin-fanta/node-http-jenkins for url <http://jenkins.jx.15>

リポジトリが出来る

 kamijin-fanta / node-http

Unwatch 1

Star 0

Fork 0

<> Code

Issues 0

Pull requests 1

Projects 0

Wiki

Insights

Settings

No description, website, or topics provided.

Edit

Add topics

3 commits

2 branches

2 releases

1 contributor

Apache-2.0

Branch: master


New pull request

Create new file

Upload files

Find file

Clone or download

 kamijin-fanta Update index.html Latest commit e9f90b5 10 days ago

charts	Draft create	10 days ago
.dockerignore	Draft create	10 days ago
.gitignore	Initial import	10 days ago
.helmignore	Draft create	10 days ago
Dockerfile	Draft create	10 days ago
Jenkinsfile	Draft create	10 days ago
LICENSE	Initial import	10 days ago

CIが回る







Jenkins

PipelinesAdministration

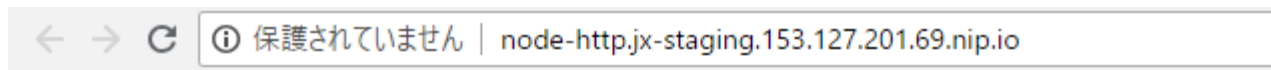
Logout

Pipelines

New Pipeline

NAME	HEALTH	BRANCHES	PR	
kamijin-fanta / environment-dukefield-production		-	-	
kamijin-fanta / environment-dukefield-staging		1 passing	-	
kamijin-fanta / node-http-jenkins		-	-	

勝手にデプロイされる



Hello Node from Jenkins X

環境一覧


- Develop
 - PRごとに作られる
- Staging
 - masterブランチが自動的にデプロイされる
- Production
 - `promote` コマンドで明示的にデプロイを行う

PRを出してみる



Update index.html #1


 Open kamijin-fanta wants to merge 1 commit into master from patch

 Conversation 1  Commits 1  Checks 0  Files changed 1

 kamijin-fanta commented 2 minutes ago Owner + 😊 ✎



No description provided.


  Update index.html Verified ✓ f6277c4

 kamijin-fanta commented just now Owner + 😊 ...

★ PR built and available in a preview environment kamijin-fanta-node-http-pr-1 [here](#)

Add more commits by pushing to the **patch** branch on kamijin-fanta/node-http.

  **All checks have passed** Show all checks
1 successful check

 **This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request ▾ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

勝手にCIが回る

✓ [kamijin-fanta / node-http 1](#)

Pipeline

Changes

Tests

Artifacts

🔄

⚙️

🔗

Logout

✕

Pull Request: PR-1 [🔗](#)

🕒 1m 47s

No changes

Commit: f6277c4

🕒 a minute ago

Pull request #1 opened



Promote to Environments



No steps This stage has no steps

勝手に開発環境にデプロイされる

← → ↻ ⓘ 保護されていません | node-http.jx-kamijin-fanta-node-http-pr-1.153.127.201.69.nip.io



Hello Node from Jenkins X

[edit from github hoge](#)

ステージング・本番

- マージ
 - masterがステージングがデプロイされる
- 本番デプロイしたい
 - `jx promote APP_NAME --version VERSION --env production`

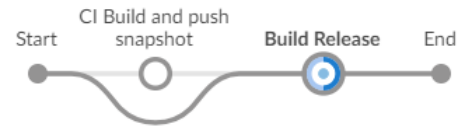
自動的にデプロイが走る

kamijin-fanta / node-http-jenkins 1

Pipeline Changes Tests Artifacts

Branch: master 2m 2s No changes

Commit: eb4aedb - Branch indexing



Build Release - 14s

✓	> Shell Script	3s
✓	> git config --global credential.helper store — Shell Script	3s
✓	> jx step validate --min-jx-version 1.1.73 — Shell Script	2s
✓	> jx step git credentials — Shell Script	3s
○	> echo \$(jx-release-version) > VERSION — Shell Script	1m 39s

- デプロイ完了
 - <http://node-http.jx-staging.153.127.201.69.nip.io/>
 - <http://node-http.jx-production.153.127.201.69.nip.io/>

jenkins-x

- デプロイの属人性の排除が出来る
- 環境毎にクラスタ分けたりも出来る
- Jenkins on Rails?

ちなみに...

利点だけではない

- カーネルを共有している
 - セキュリティ
 - パフォーマンス
- CPU依存(マルチアーキテクチャはあるが...)
- クラスタの管理・アップグレード

まとめ

- Dockerでデプロイ環境を作ること、開発から本番まで一貫した環境を用意できる
- Docker周辺技術には様々な選択肢が用意されている
 - プロジェクトの要件・規模に応じて選択を

おわり