Docker for Developers

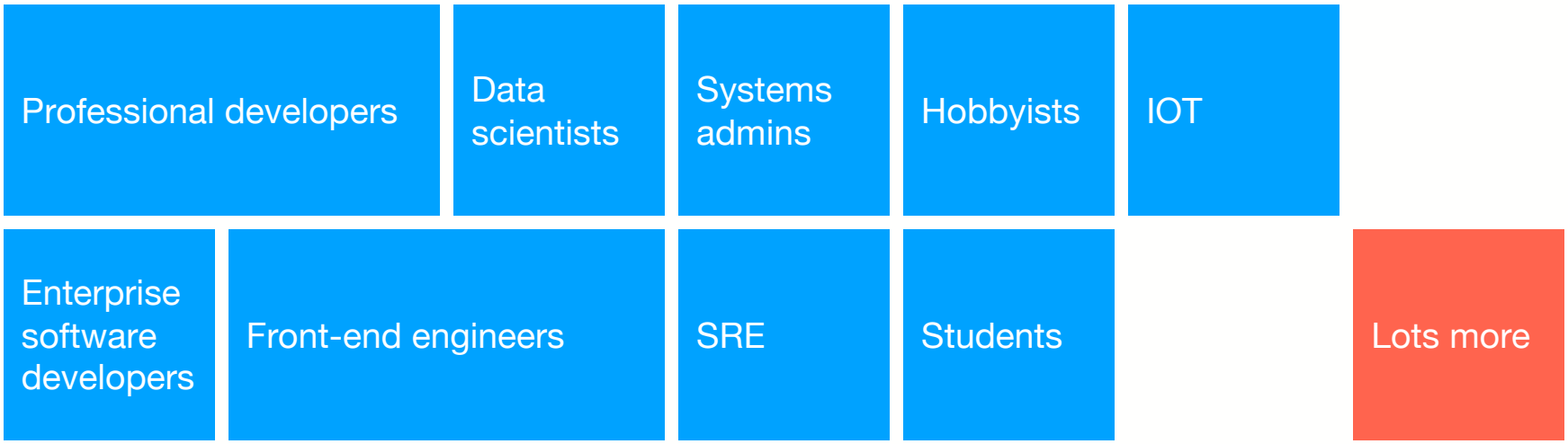
GARETH RUSHGROVE
Product Manager, Docker

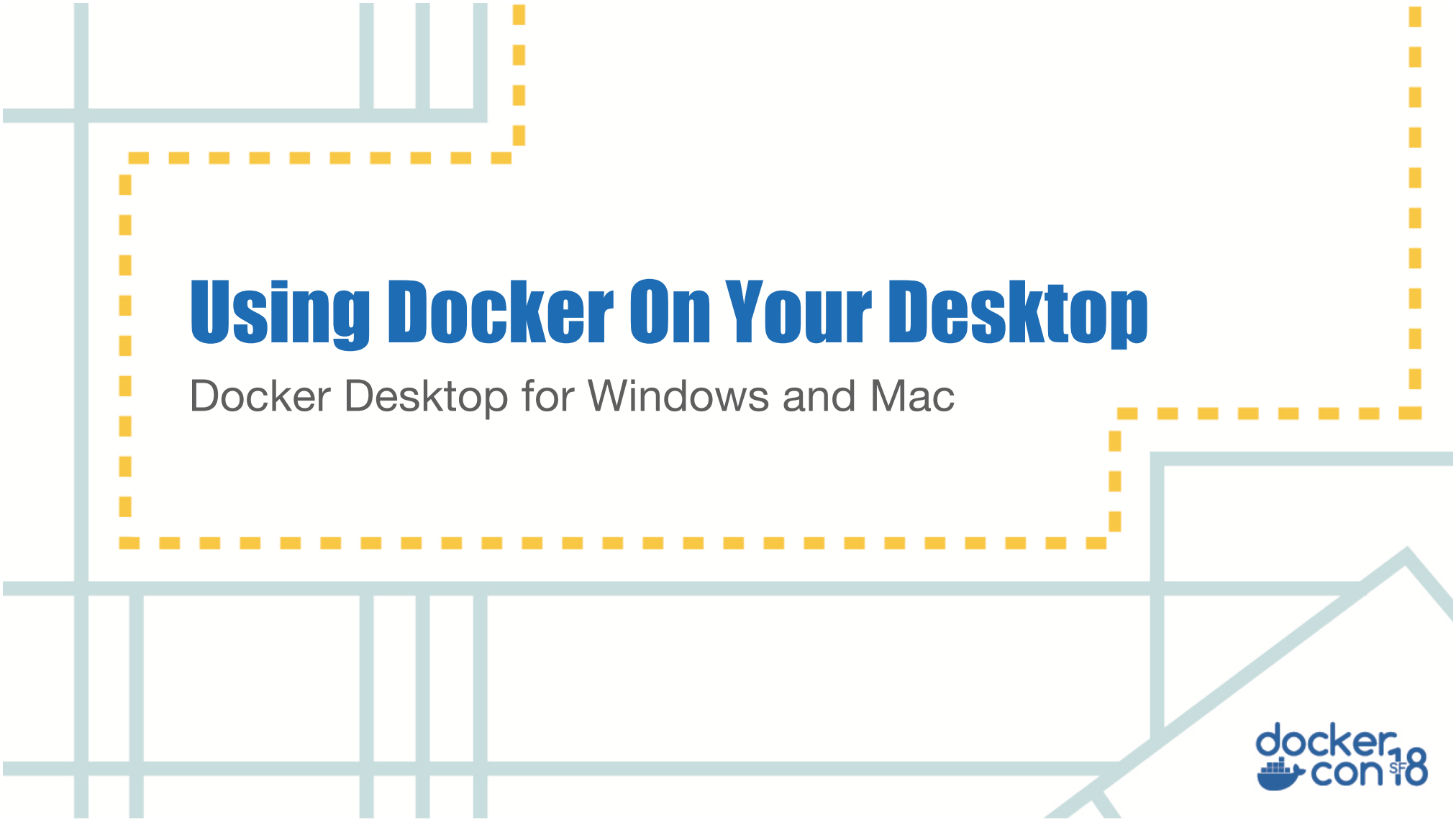
dockercon SF 18

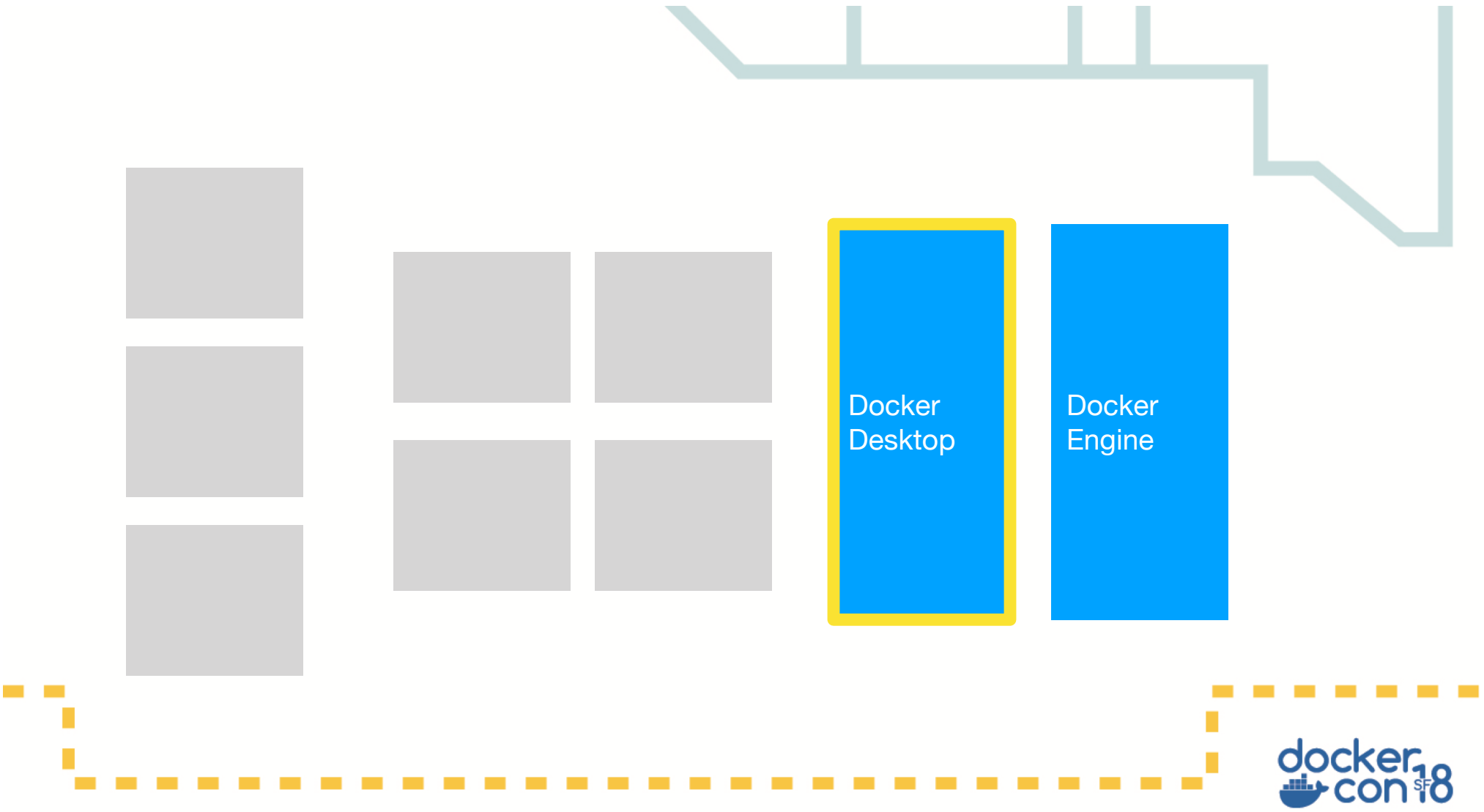
This Talk

- Using Docker on your desktop
- Building Docker images
- Describing applications in code
- Using IDE's with Docker
- Graphical tools for creating applications

What Do We Mean By Developers?




Using Docker On Your Desktop
Docker Desktop for Windows and Mac
docker con SF 18


Docker
Desktop
Docker
Engine

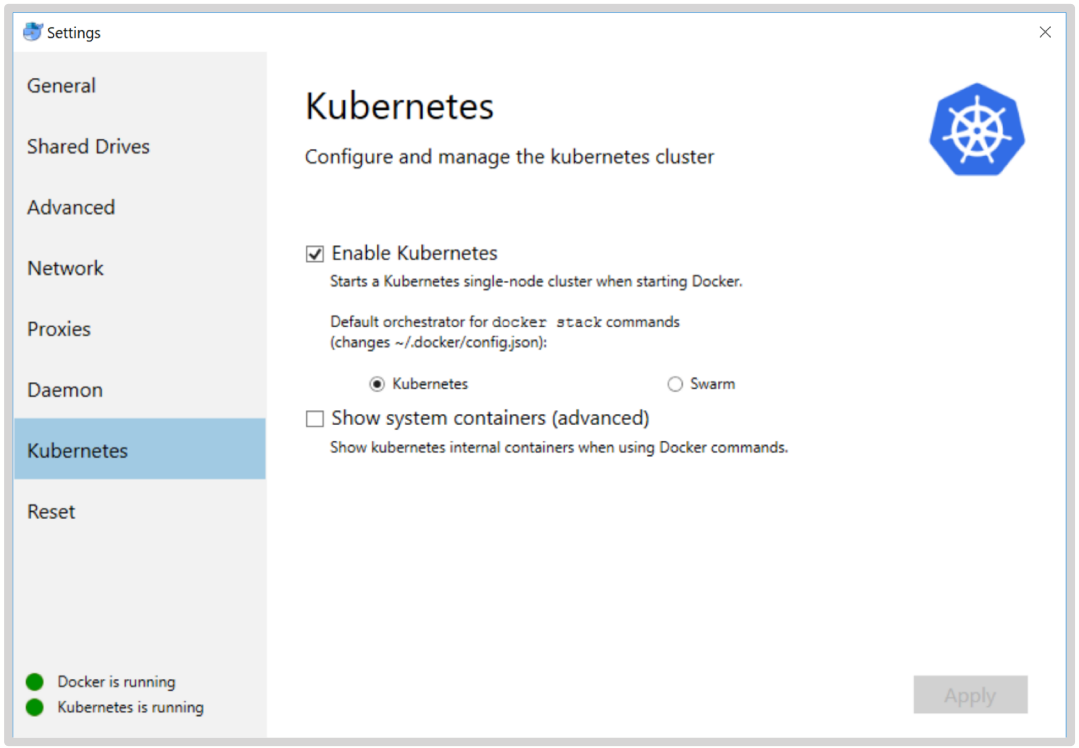
Docker On Your Desktop

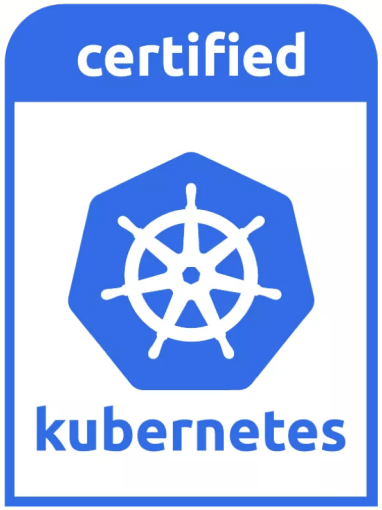
- **Avoid “works on my machine”**
  Fix issues locally, before they hit production
- **Improve productivity**
  Lots of tools already built for Docker
- **Speed up project onboarding**
  Start with just a Compose file and Docker Desktop

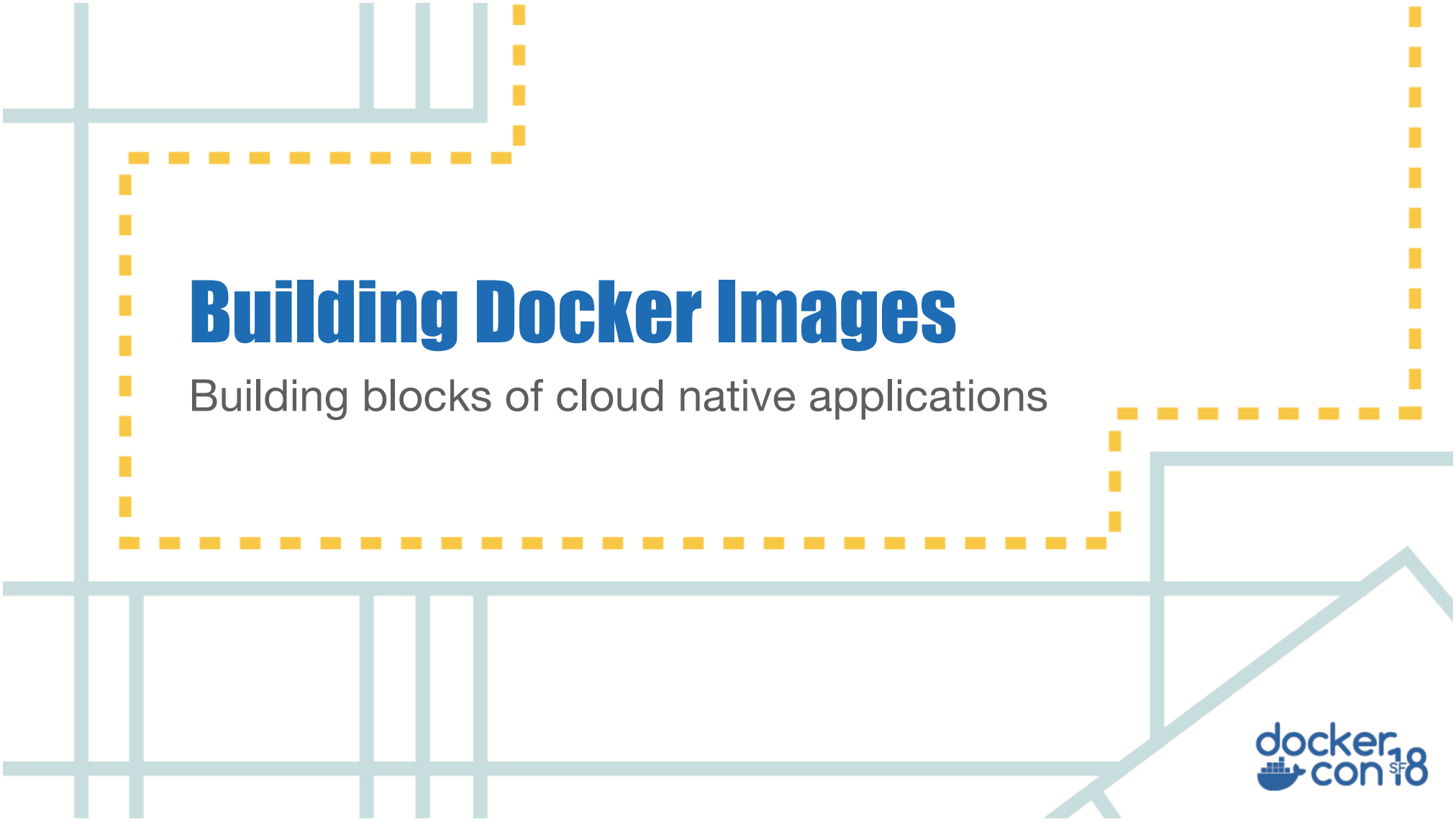
Optimised For Developers

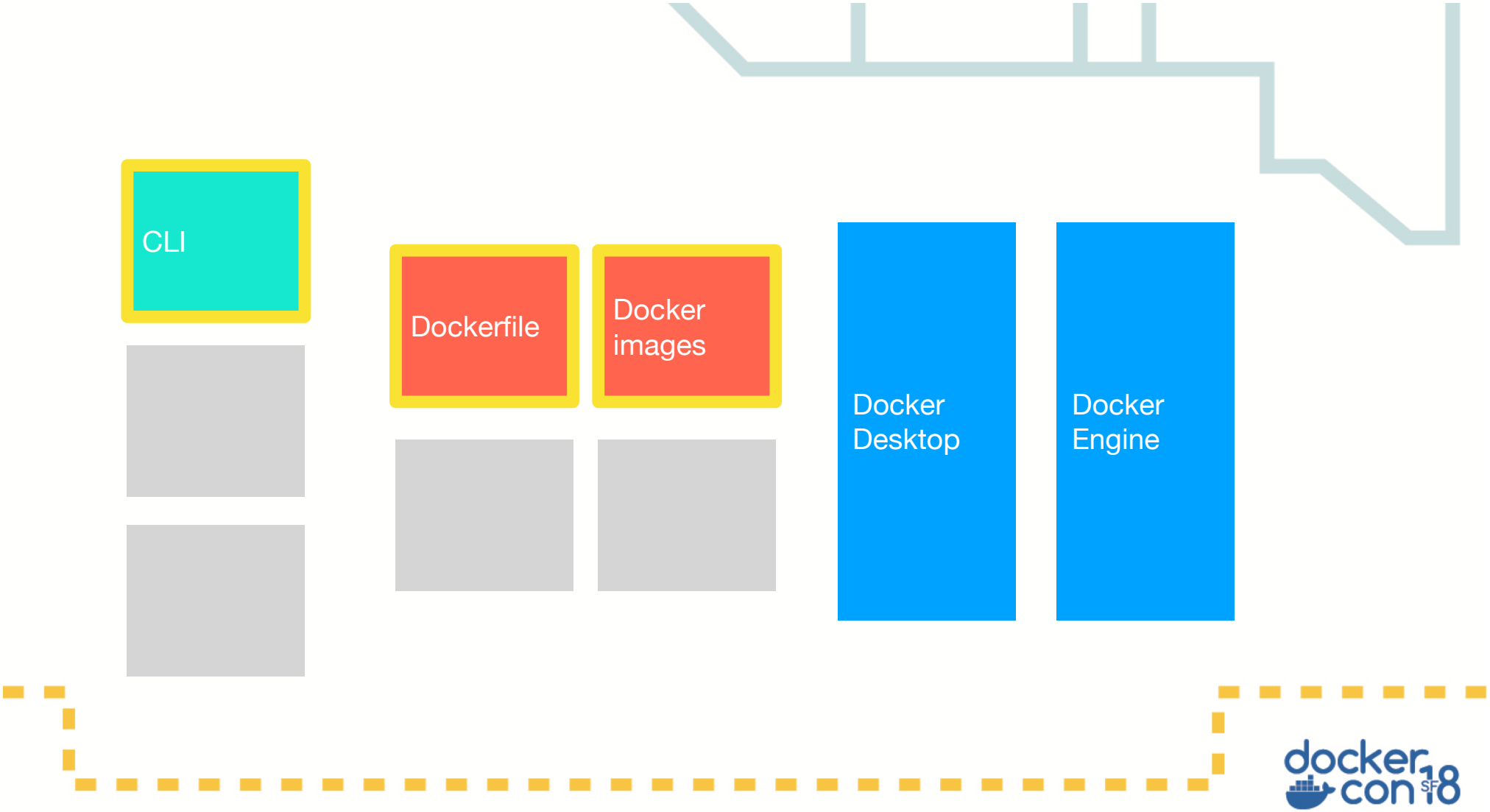
- **Lightweight managed virtual machine**
  Small overhead and stays out of your way
- **File system optimisations**
  Lots of work on improving performance for shared folders
- **Native networking**
  Access services on localhost with no fuss

Kubernetes On Your Desktop






Building Docker Images
Building blocks of cloud native applications
docker con SF 18


CLI
Dockerfile
Docker images
Docker Desktop
Docker Engine

Dockerfile!

```
# Use an official Python runtime as a parent image
FROM python:2.7-slim
# Set the working directory to /app
WORKDIR /app
# Copy the current directory contents into the container at /app
ADD . /app
# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt
# Make port 80 available to the world outside this container
EXPOSE 80
# Run app.py when the container launches
CMD ["python", "app.py"]
```

Why Dockerfile?

- **Simple text format**
  Include alongside your source code in version control
- **One way of packaging any app**
  Works for any language or framework
- **Already familiar to lots of developers**
  Run commands you already know

Dockerfile to Docker Image

```
$ docker build -t friendlyhello .
```

```
$ docker image ls

REPOSITORY          TAG                 IMAGE ID
friendlyhello       latest              326387cea398
```

Shell Scripts and the Builder Pattern

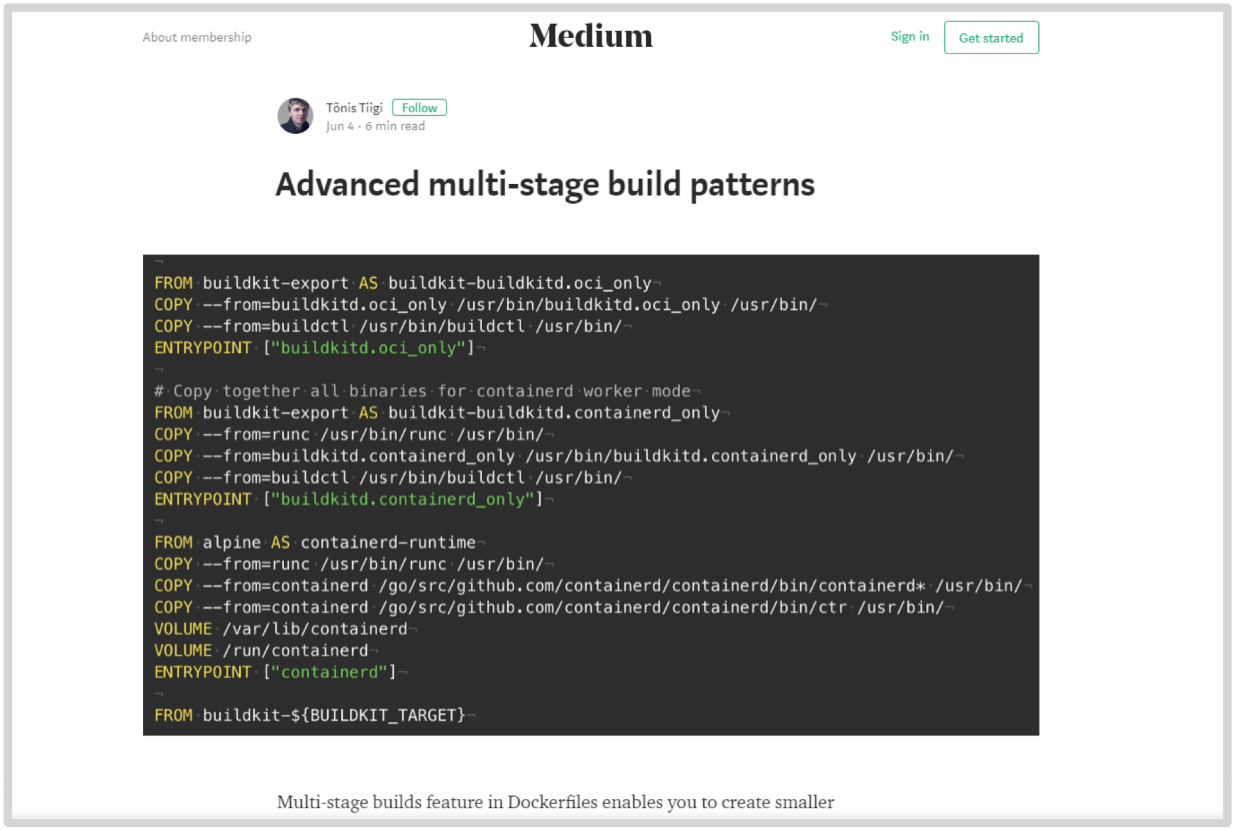
```
#!/bin/sh
echo Building alexellis2/href-counter:build

docker build --build-arg https_proxy=$https_proxy --build-arg http_proxy=$http_proxy \
    -t alexellis2/href-counter:build . -f Dockerfile.build

docker container create --name extract alexellis2/href-counter:build
docker container cp extract:/go/src/github.com/alexellis/href-counter/app ./app
docker container rm -f extract

echo Building alexellis2/href-counter:latest

docker build --no-cache -t alexellis2/href-counter:latest .
rm ./app
```

No Longer Required

Multi-stage Builds!

```
FROM golang:1.7.3
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=0 /go/src/github.com/alexellis/href-counter/app .
CMD ["./app"]
```

Advanced Multi-stage Build Patterns



About membership

Medium

Sign in Get started

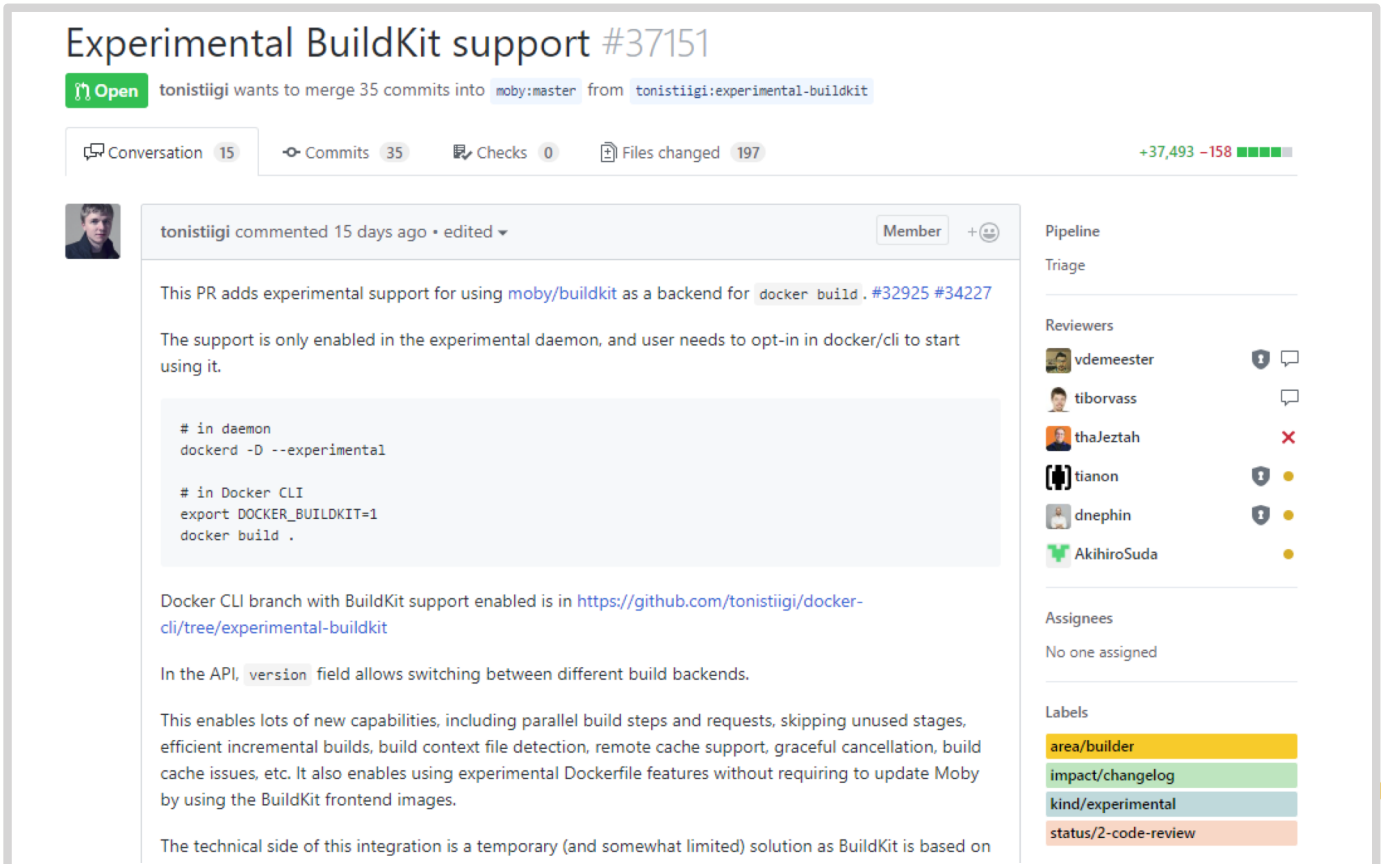
Tõnis Tiigi Follow
Jun 4 · 6 min read

Advanced multi-stage build patterns

```
FROM buildkit-export AS buildkit-buildkitd.oci_only
COPY --from=buildkitd.oci_only /usr/bin/buildkitd.oci_only /usr/bin/
COPY --from=buildctl /usr/bin/buildctl /usr/bin/
ENTRYPOINT ["buildkitd.oci_only"]

# Copy together all binaries for containerd worker mode
FROM buildkit-export AS buildkit-buildkitd.containerd_only
COPY --from=runc /usr/bin/runc /usr/bin/
COPY --from=buildkitd.containerd_only /usr/bin/buildkitd.containerd_only /usr/bin/
COPY --from=buildctl /usr/bin/buildctl /usr/bin/
ENTRYPOINT ["buildkitd.containerd_only"]

FROM alpine AS containerd-runtime
COPY --from=runc /usr/bin/runc /usr/bin/
COPY --from=containerd /go/src/github.com/containerd/containerd/bin/containerd* /usr/bin/
COPY --from=containerd /go/src/github.com/containerd/containerd/bin/ctr /usr/bin/
VOLUME /var/lib/containerd
VOLUME /run/containerd
ENTRYPOINT ["containerd"]

FROM buildkit-${BUILDKIT_TARGET}
```

Multi-stage builds feature in Dockerfiles enables you to create smaller

BuildKit Support is Coming



Experimental BuildKit support #37151

Open **tonistiigi** wants to merge 35 commits into `moby:master` from `tonistiigi:experimental-buildkit`

Conversation 15 · Commits 35 · Checks 0 · Files changed 197 · +37,493 −158

tonistiigi commented 15 days ago • edited — Member

This PR adds experimental support for using moby/buildkit as a backend for `docker build`. #32925 #34227

The support is only enabled in the experimental daemon, and user needs to opt-in in docker/cli to start using it.

```
# in daemon
dockerd -D --experimental

# in Docker CLI
export DOCKER_BUILDKIT=1
docker build .
```

Docker CLI branch with BuildKit support enabled is in https://github.com/tonistiigi/docker-cli/tree/experimental-buildkit

In the API, `version` field allows switching between different build backends.

This enables lots of new capabilities, including parallel build steps and requests, skipping unused stages, efficient incremental builds, build context file detection, remote cache support, graceful cancellation, build cache issues, etc. It also enables using experimental Dockerfile features without requiring to update Moby by using the BuildKit frontend images.

The technical side of this integration is a temporary (and somewhat limited) solution as BuildKit is based on

Pipeline
Triage

Reviewers
vdemeester
tiborvass
thaJeztah
tianon
dnephin
AkihiroSuda

Assignees
No one assigned

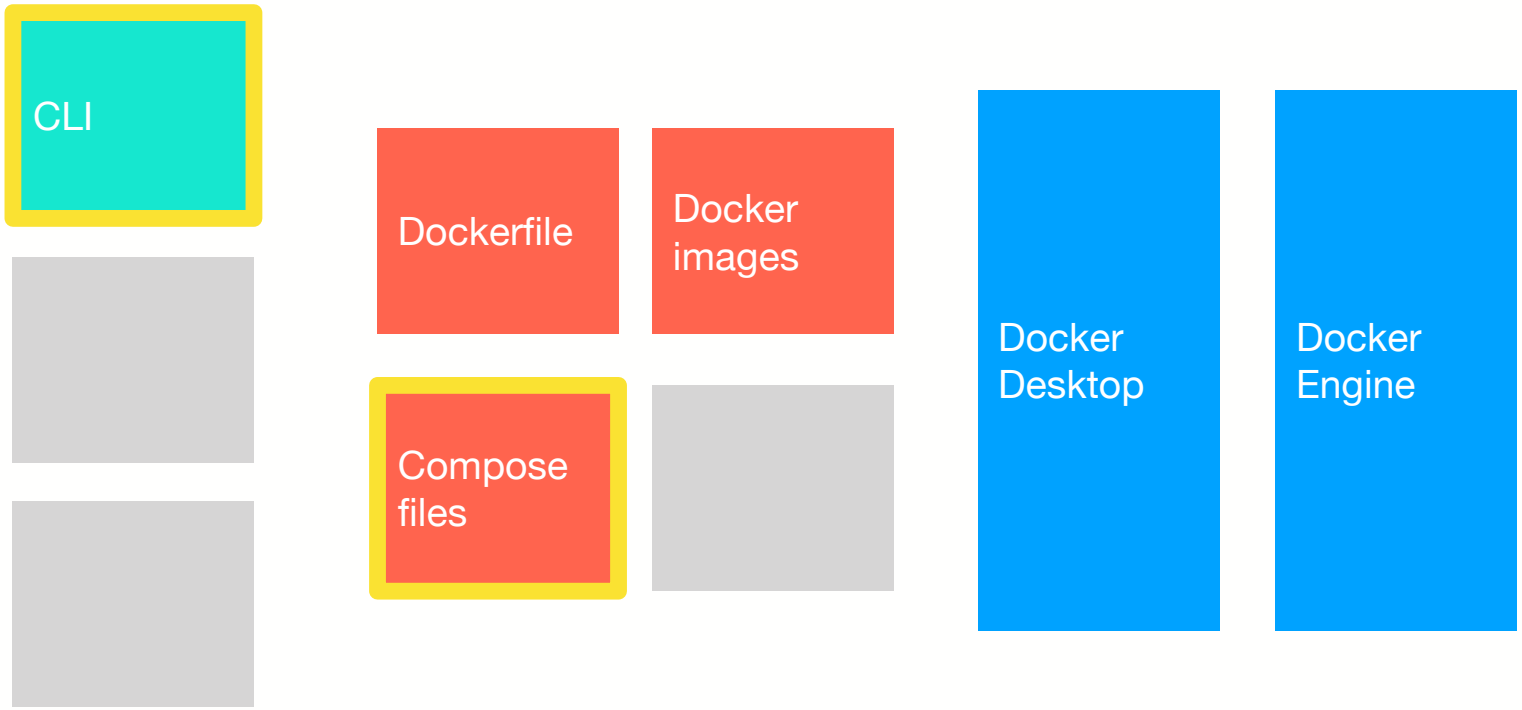
Labels
area/builder
impact/changelog
kind/experimental
status/2-code-review

Things to Look Forward to With BuildKit

- Faster builds
- Garbage collection of build artefacts
- Parallel builds of multiple images
- Remote cache support
- Custom Dockerfile syntax
- Entire new build frontends

Describing Applications in Code

Docker Compose and friends


CLI
Dockerfile
Docker images
Compose files
Docker Desktop
Docker Engine

Compose Files!

```
version: '3.6'

services:
  db:
    image: postgres
  web:
    build: .
    command: python3 manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db
```

Run Services Described in Compose

```
$ docker-compose up
djangosample_db_1 is up-to-date
Creating djangosample_web_1 ...
Creating djangosample_web_1 ... done
Attaching to djangosample_db_1, djangosample_web_1
db_1   | The files belonging to this database system will be owned by user "postgres".
db_1   | This user must also own the server process.
db_1   |
db_1   | The database cluster will be initialized with locale "en_US.utf8".
db_1   | The default datweb_1  | May 30, 2017 - 21:44:49
db_1   | The default text search configuration will be set to "english".
web_1  | Django version 1.11.1, using settings 'composeexample.settings'
web_1  | Starting development server at http://0.0.0.0:8000/
web_1  | Quit the server with CONTROL-C.abase encoding has accordingly been set to "UTF8".
```

Why Compose Files?

- **Simple text format**
  Include alongside your source code in version control
- **Most applications consist of multiple services**
  Microservices or even just a supporting database
- **One command to start all dependencies**
  Get up and running with a new application quickly

Compose Works on Kubernetes

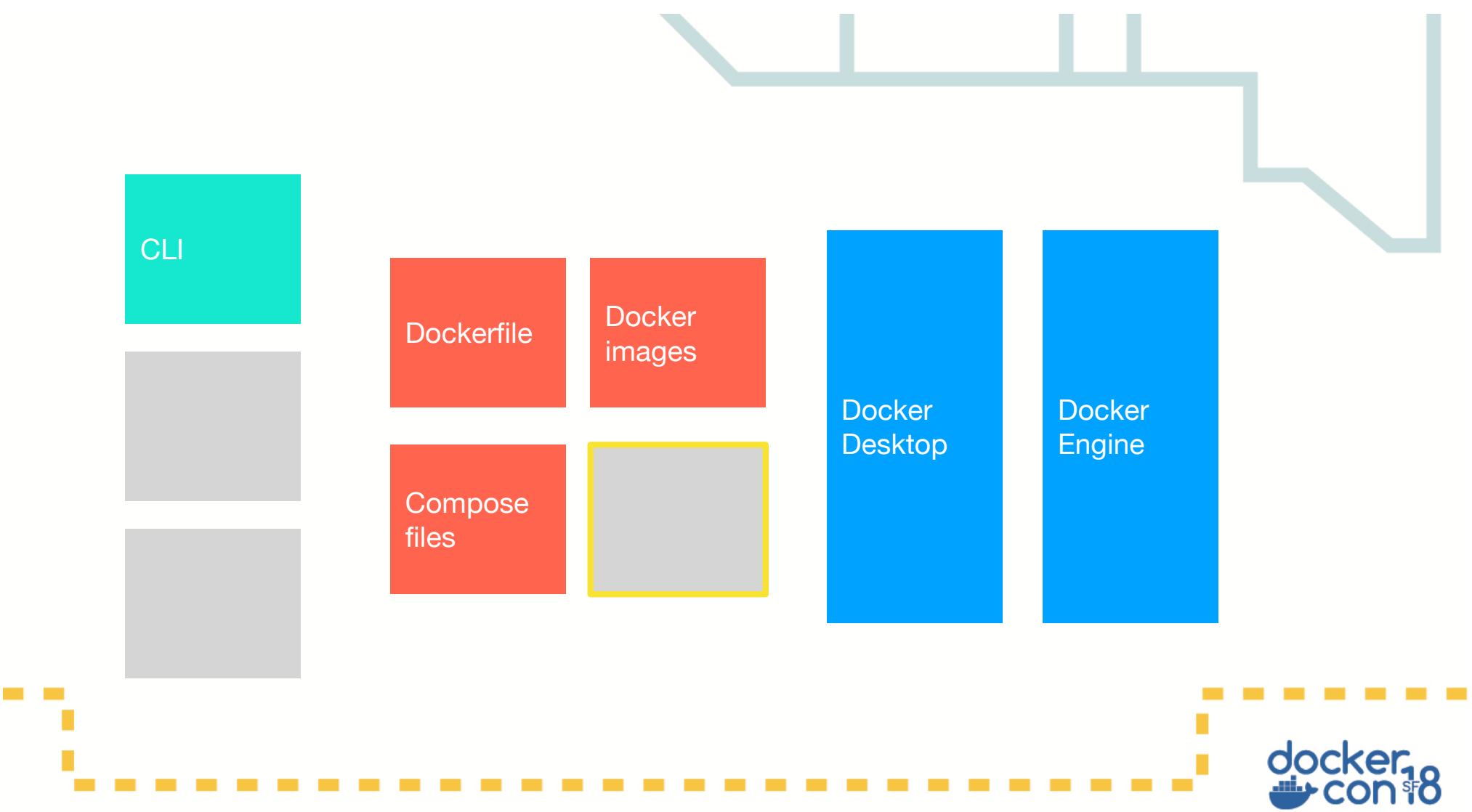
```
$ docker stack deploy --compose-file words.yaml words
Stack words was created
Waiting for the stack to be stable and running...
 - Service db has one container running
 - Service words has one container running
 - Service web has one container running
Stack words is stable and running
```

Custom API Server Adds Stack

```
$ kubectl get deployment
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
db        1         1         1            1           2m
web       1         1         1            1           2m
words     5         5         5            5           2m
```

Why Compose on Kubernetes?

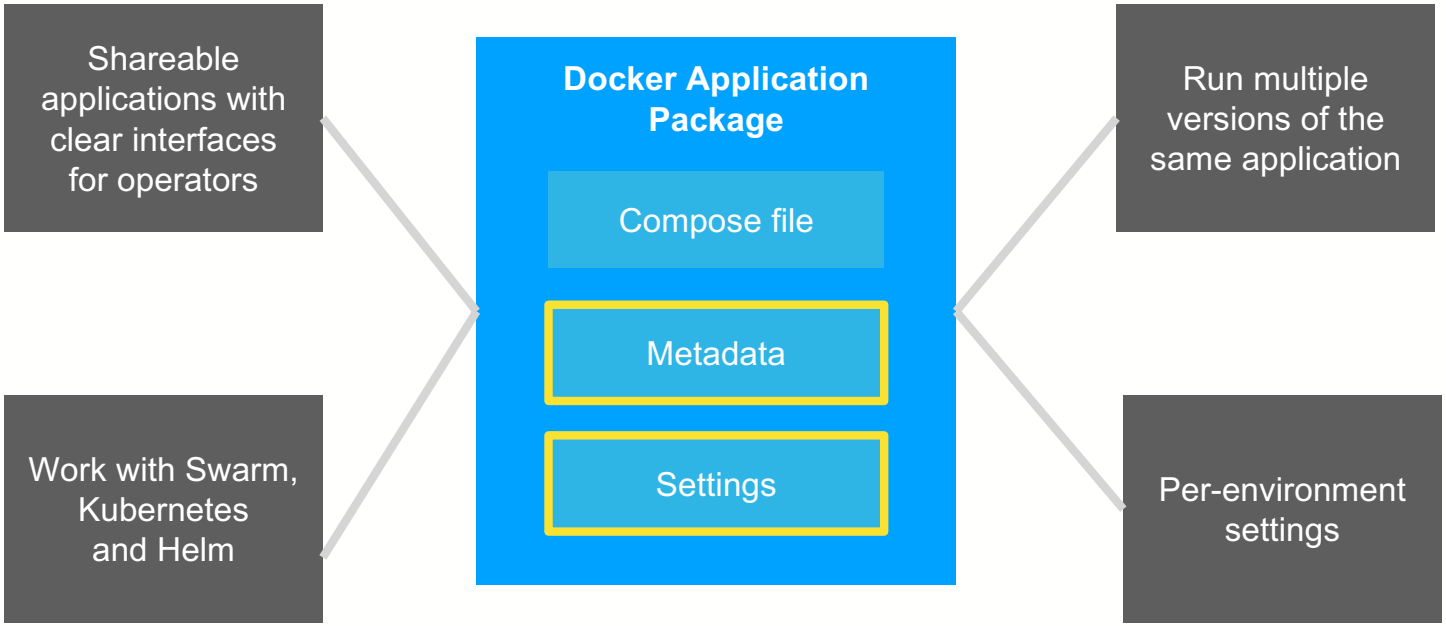
- **A higher level description**
  Optimised for a very common use cases
- **Maintain less code**
  Less verbose than the raw API
- **Portability**
  Usable with Swarm, Kubernetes or just the engine


CLI
Dockerfile
Docker images
Compose files
Docker Desktop
Docker Engine

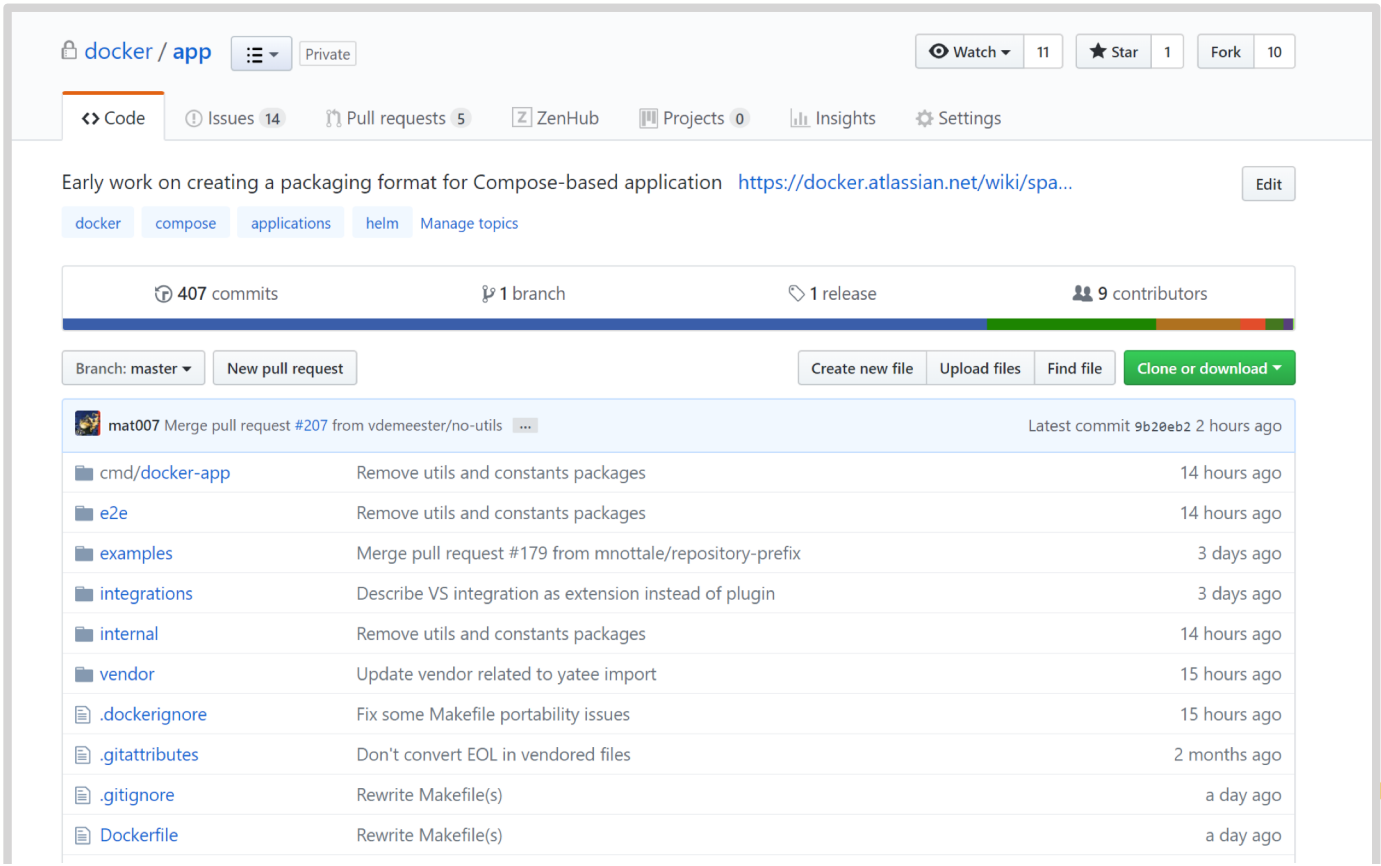
Areas to Improve

- **Hard to create reusable Compose files**
  Most reuse is via copy and paste
- **No clear interface between dev and ops**
  Hard to separate different concerns in one file
- **No central place to share Compose applications**
  Docker images are shared via a repository like Hub

Experiment: Application Packages



docker/app Now Open Source



Demo time

Create Packages From Compose

```
$ ls
docker-compose.yml
$ docker-app init hello
$ ls
docker-compose.yml
hello.dockerapp
```

Define Clear Interfaces

```
$ docker-app inspect
hello 0.1.0
Maintained by: garethr

Setting Default
------- -------
port    8080
text    hello world
version latest
```

Override Settings at Runtime

```
$ docker-app render --set port=9090
$ docker-app render -f production.yaml
...
```

Deploy Applications

```
$ docker-app deploy --set port=9090
$ docker-app deploy --orchestrator kubernetes
$ docker-app deploy -f production.yml
```

Share Applications on Hub and DTR

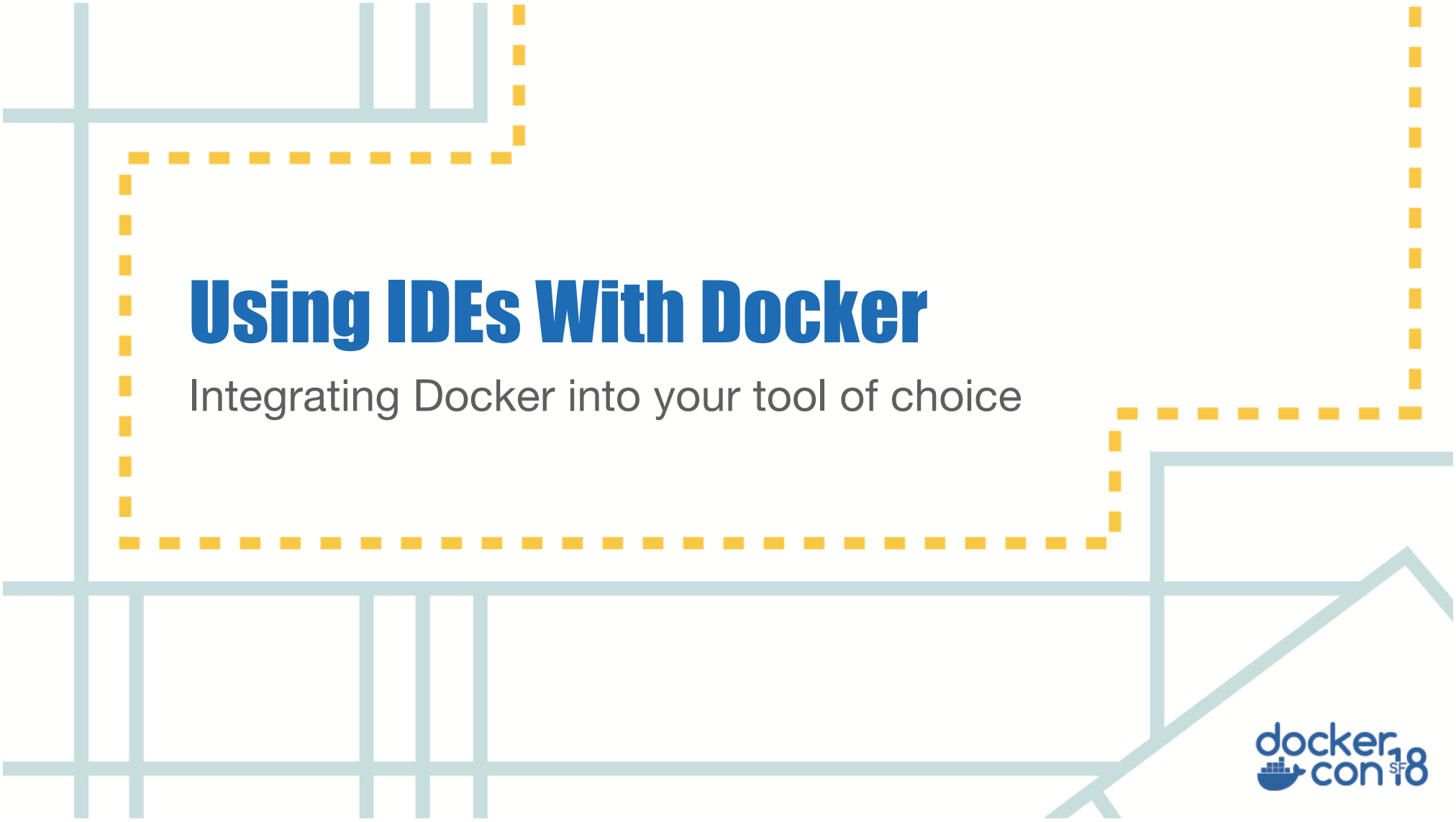
```
$ ls
hello.dockerapp
$ docker-app save
$ docker inspect hello.dockerapp
...
$ docker-app push --namespace garethr
$ docker-app deploy garethr/hello.dockerapp
```

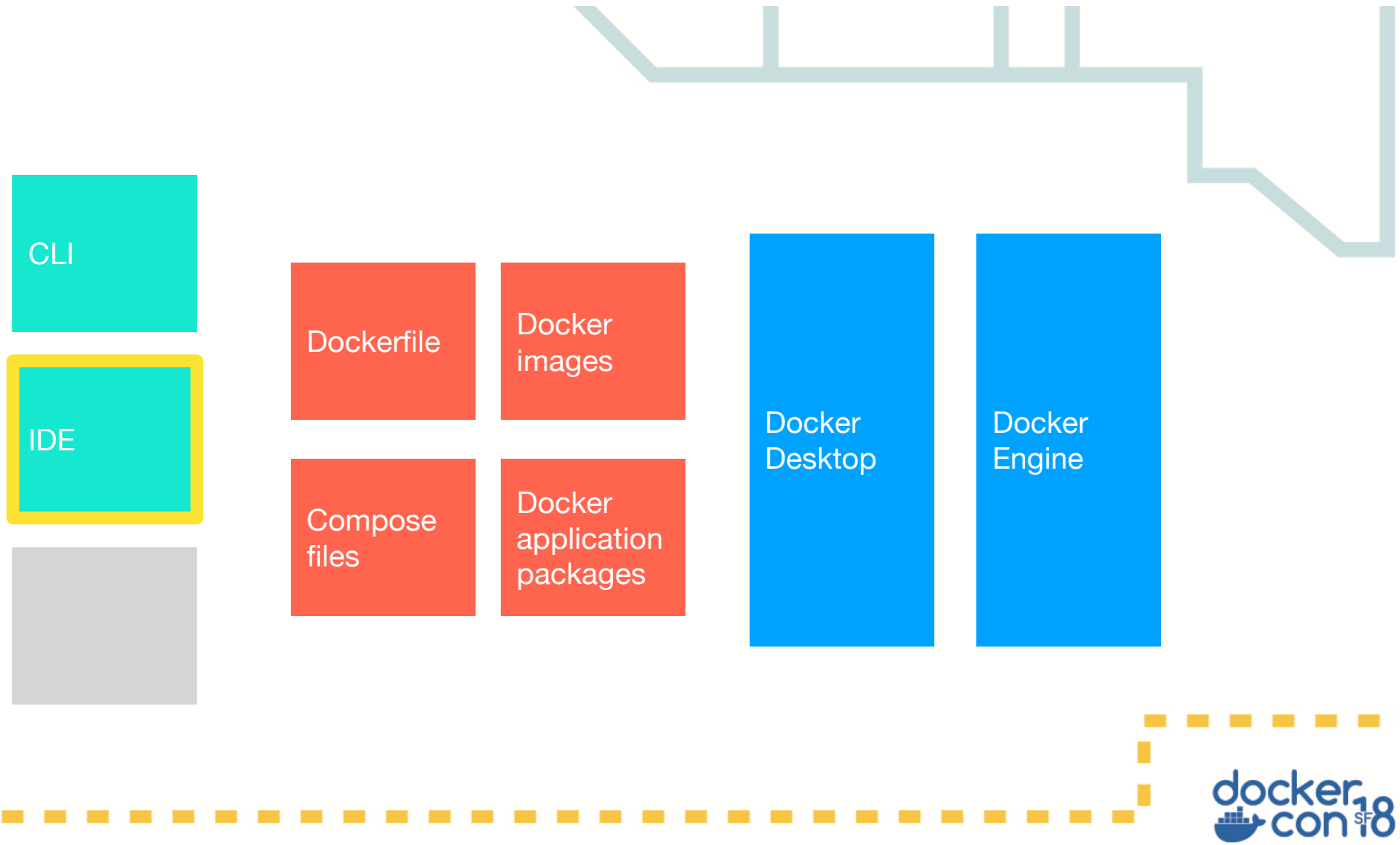
Deploy Applications Using Helm

```
$ ls
hello.dockerapp
$ docker-app helm
$ helm deploy hello.chart
```

Current Status

- **Experimental**
  Tell us what you think
- **Standalone utility (for now)**
  Allow us to move quickly and break things
- **Open source**
  Have an idea? Come and hack on the code with us


Using IDEs With Docker
Integrating Docker into your tool of choice
docker con SF 18


CLI
IDE
Dockerfile
Docker images
Compose files
Docker application packages
Docker Desktop
Docker Engine

Most Popular Editors

34.9% Visual Studio Code

34.3% Visual Studio

34.2% Notepad++

28.9% Sublime Text

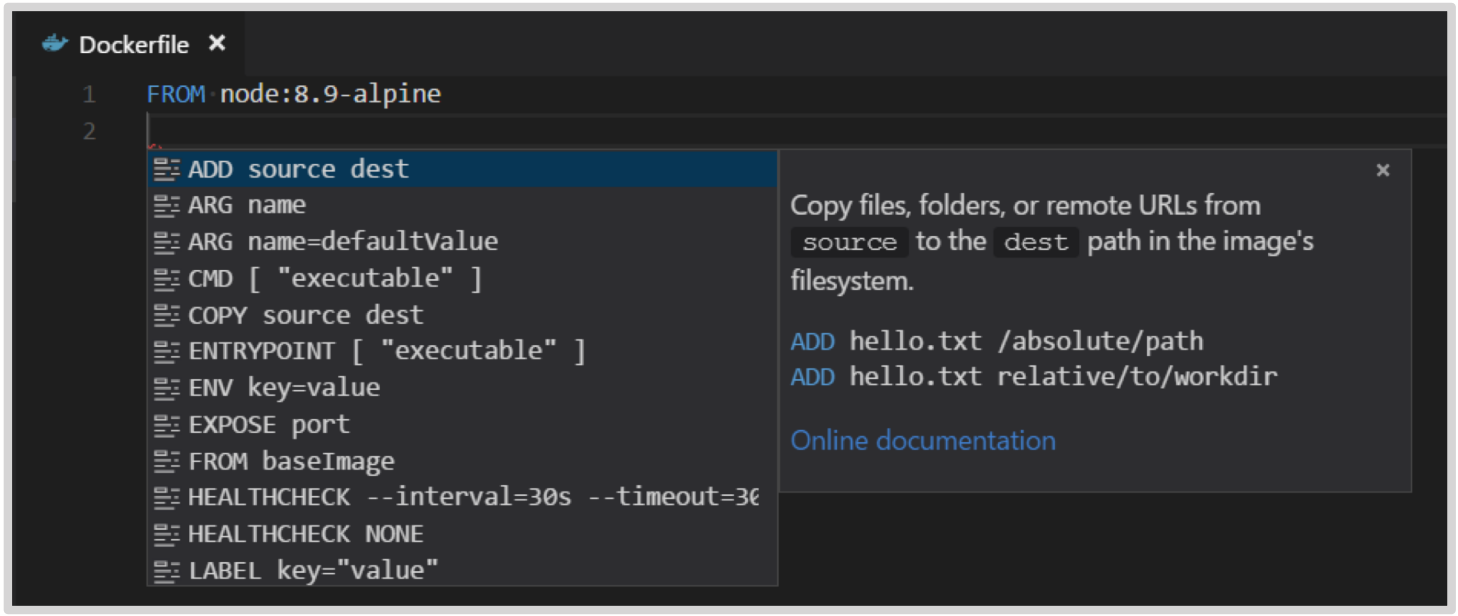
25.8% Vim

24.9% IntelliJ

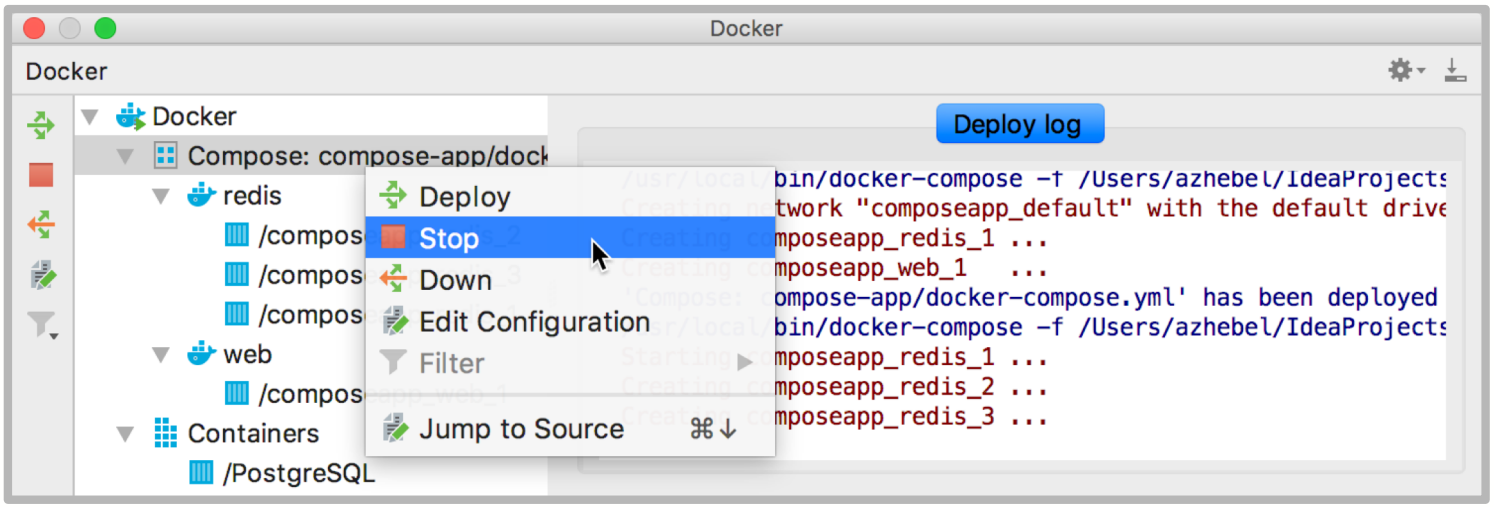
Syntax Highlighting

```
ARG ALPINE_VERSION=3.7
ARG GO_VERSION=1.10.2

FROM golang:${GO_VERSION}-alpine${ALPINE_VERSION} AS build
ARG DOCKERCLI_VERSION=18.03.1-ce
ARG DOCKERCLI_CHANNEL=edge
RUN apk add --no-cache \
  bash \
  make\
  git \
  curl \
  util-linux
RUN curl -Ls https://download.docker.com/linux/static/$DOCKERCLI_CHANNEL/x86_64/docker-$DOCKERCLI_VERSION.tgz | \
  tar -xz docker/docker && \
  ls -l . && \
  mv docker/docker /usr/bin/docker

WORKDIR /go/src/github.com/docker/app/

FROM build AS dev
COPY . .

FROM dev AS docker-app
ARG COMMIT=unknown
ARG TAG=unknown
ARG BUILDTIME=unknown
RUN make COMMIT=${COMMIT} TAG=${TAG} BUILDTIME=${BUILDTIME} bin/docker-app

```

Contextual Help

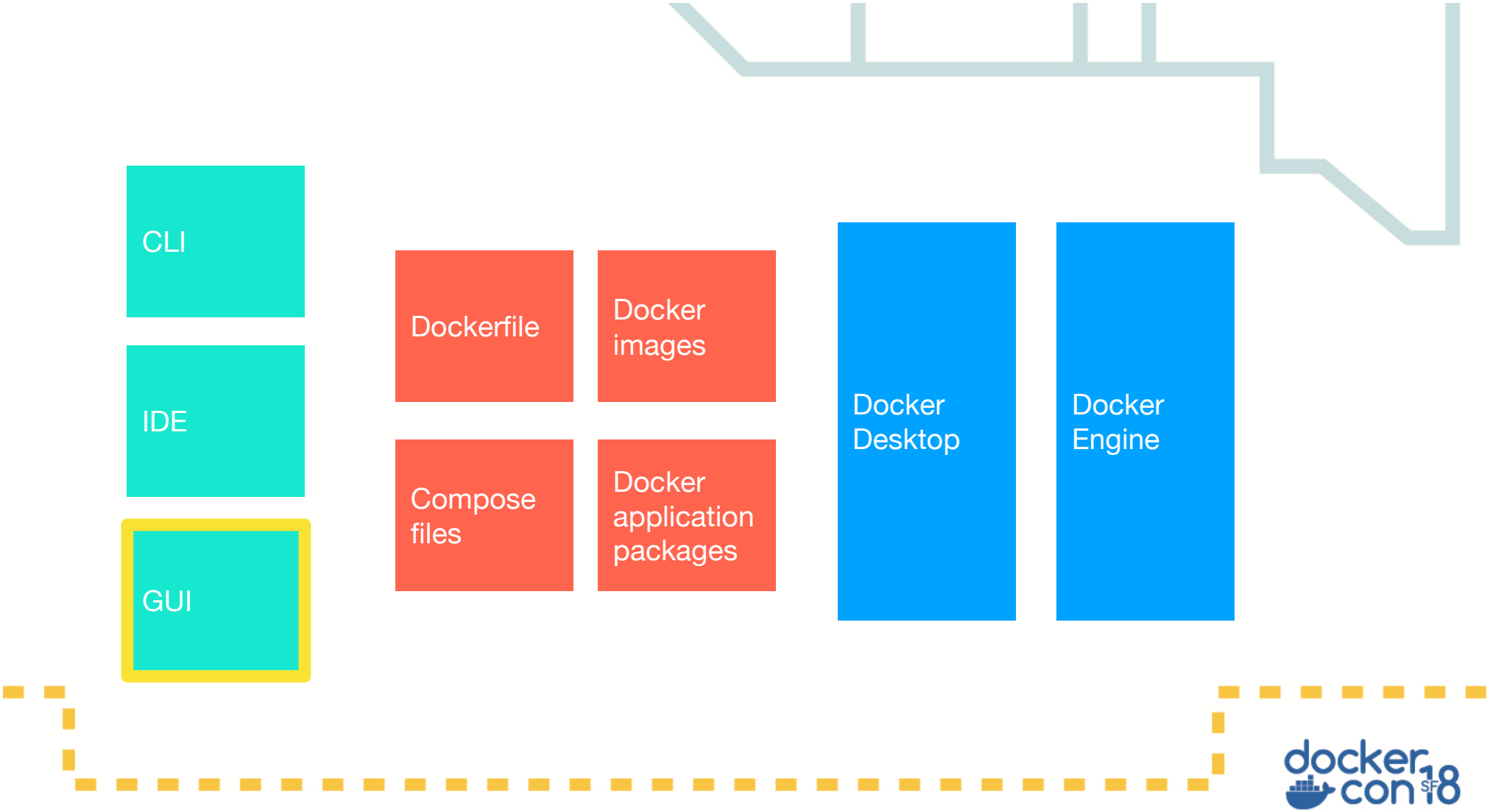


Running Containers from your IDE

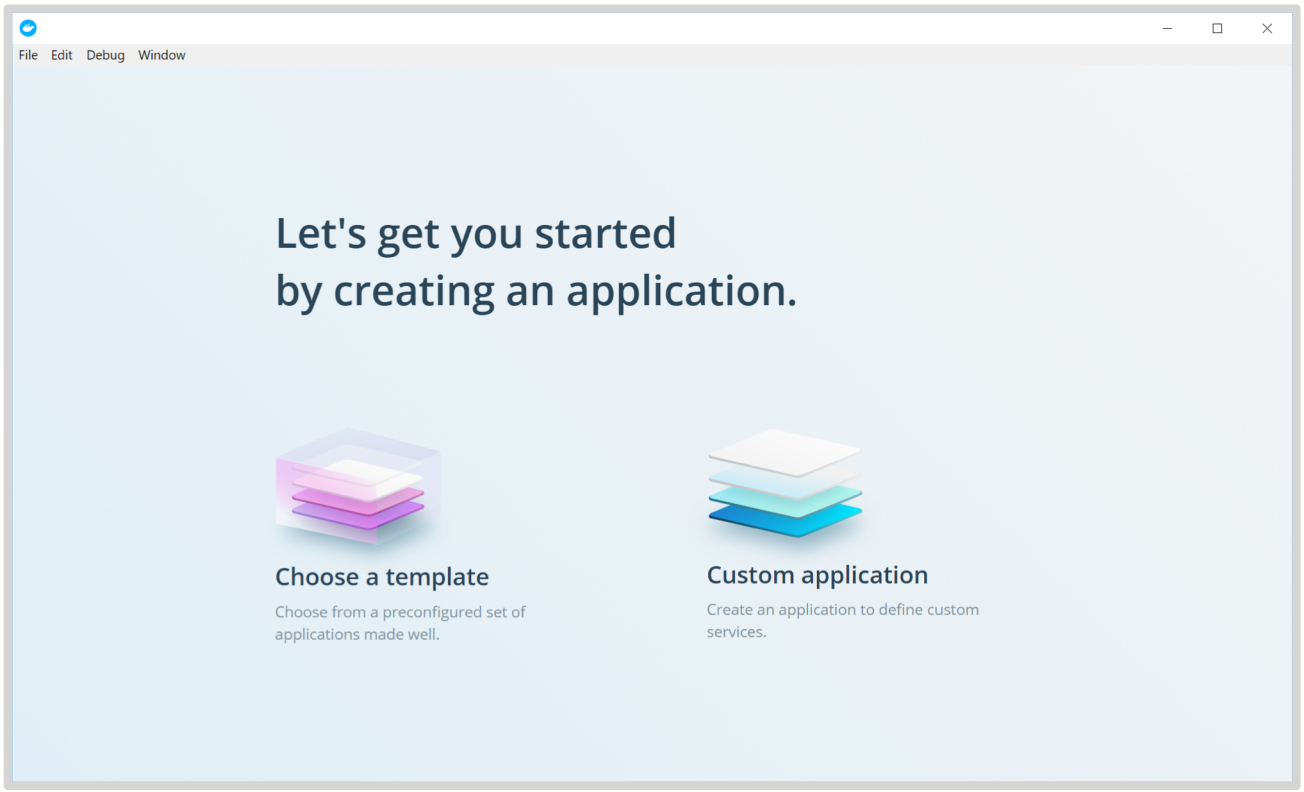


New Graphical Tools

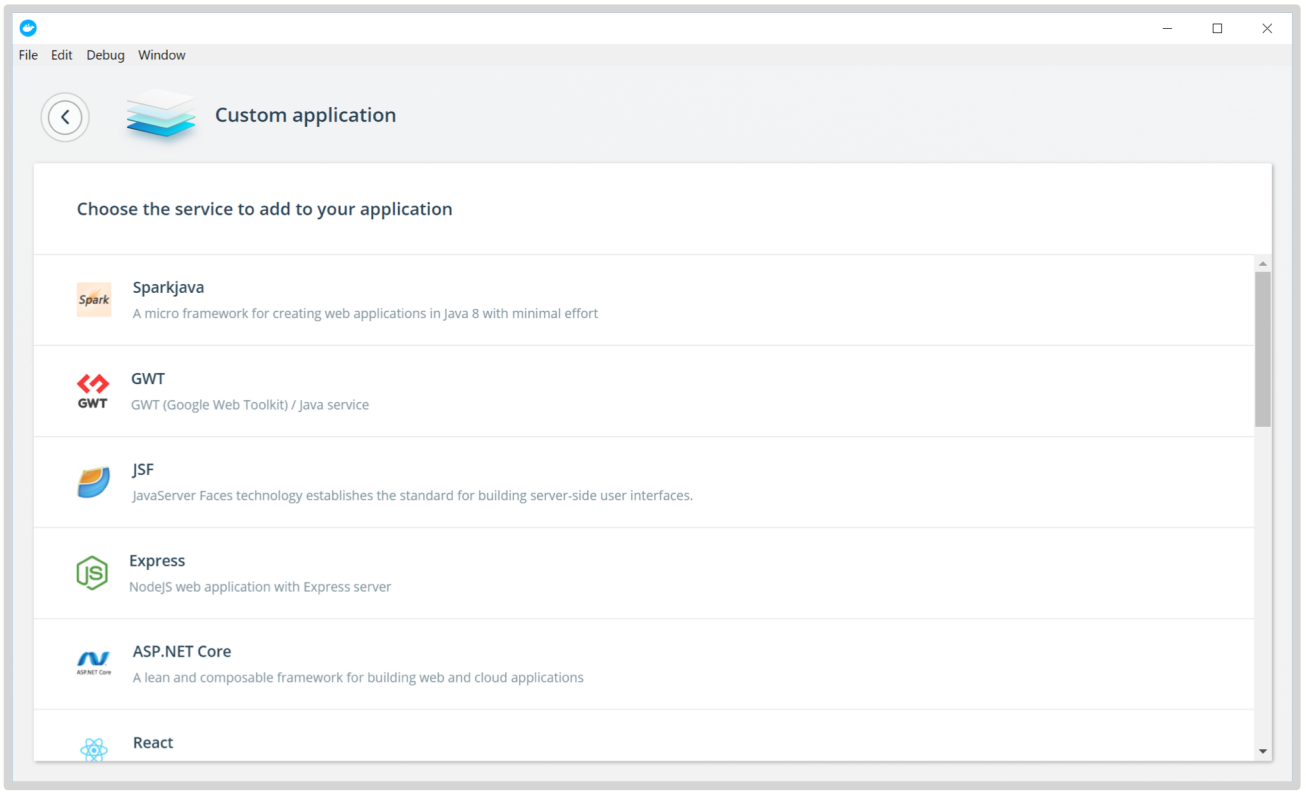
Design applications directly in Docker Desktop


CLI
IDE
GUI
Dockerfile
Docker images
Compose files
Docker application packages
Docker Desktop
Docker Engine

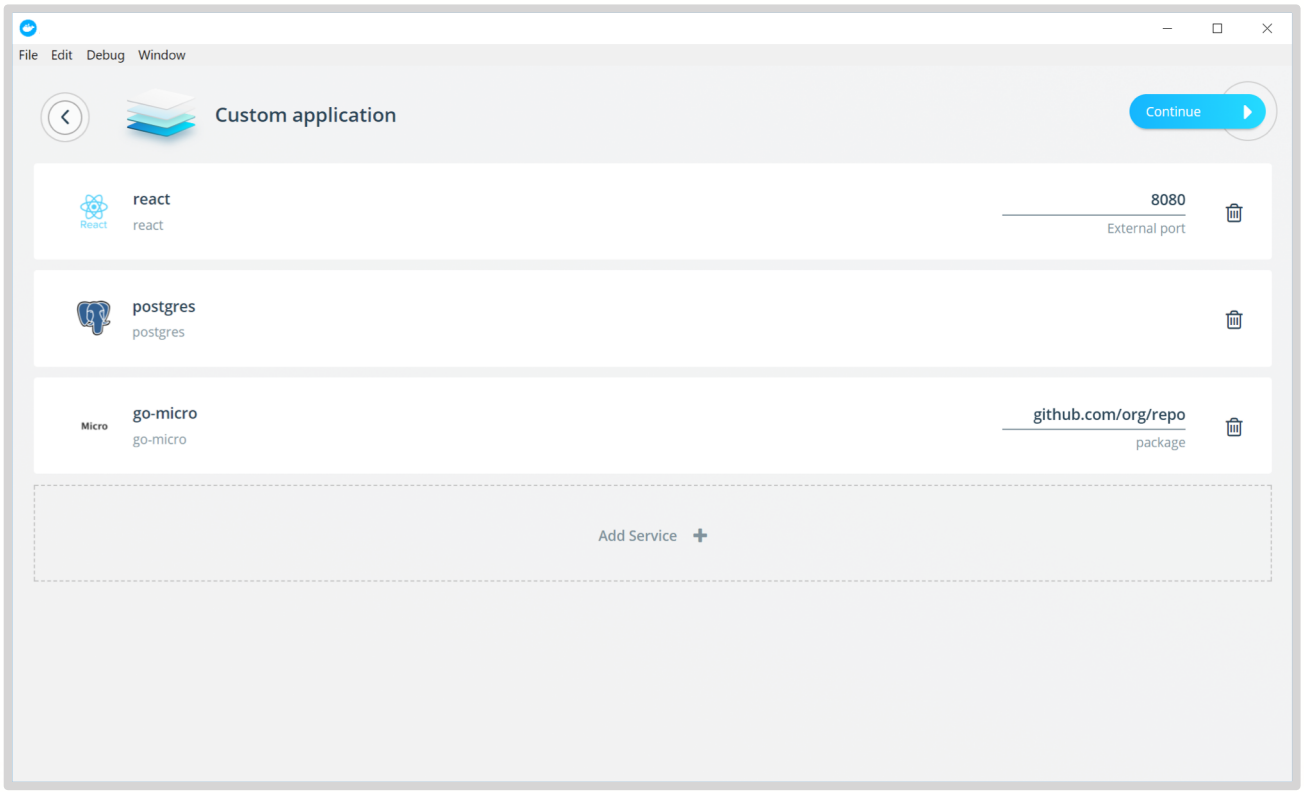
Create Applications in Docker Desktop



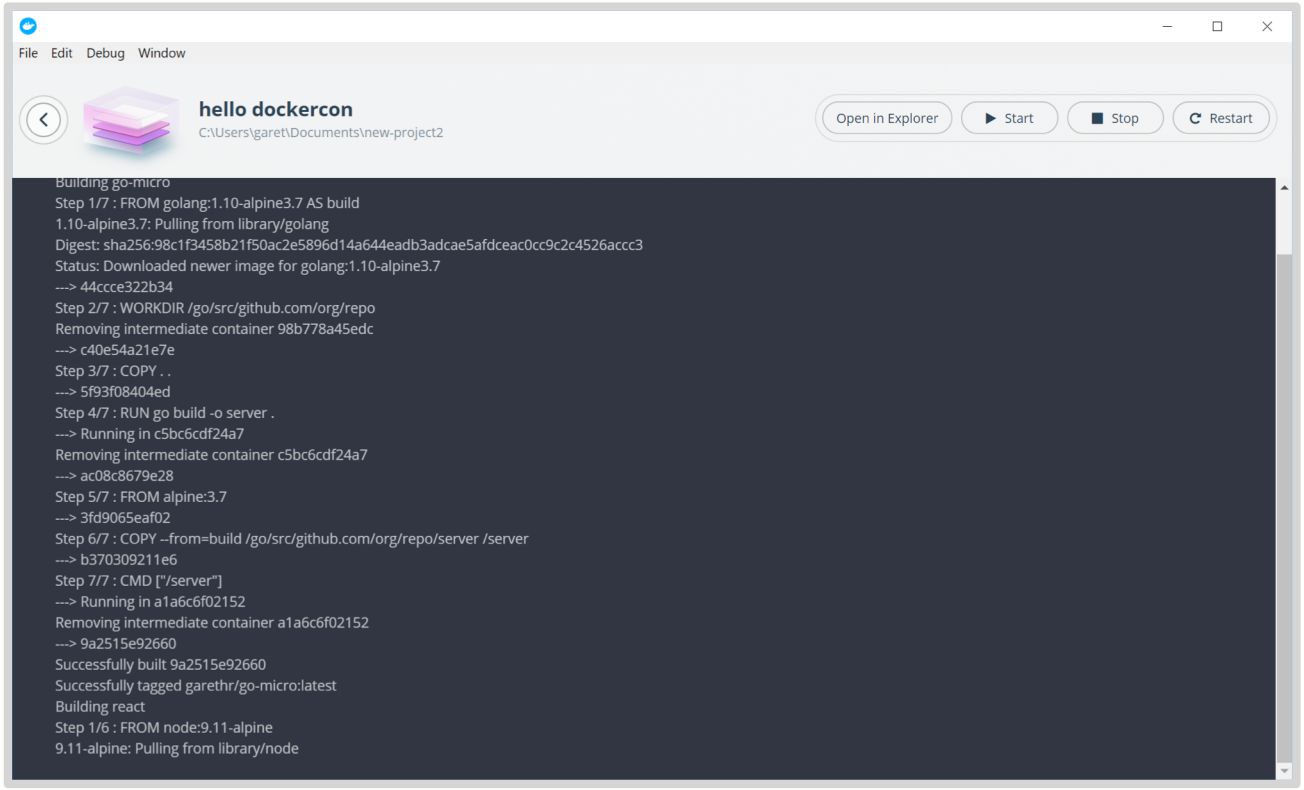
Add Services to Your Application



Configure Services



Run Your Finished Application

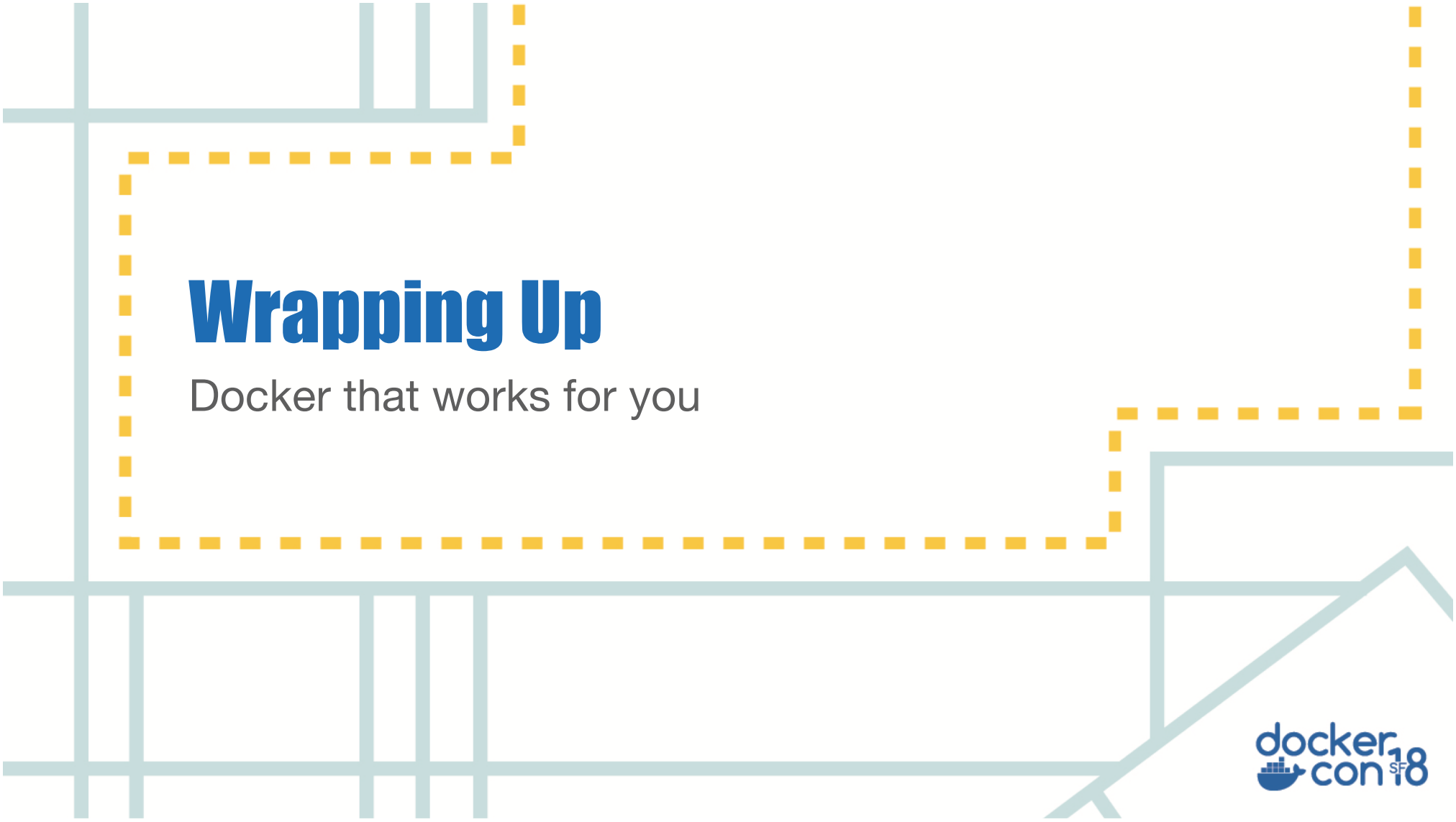


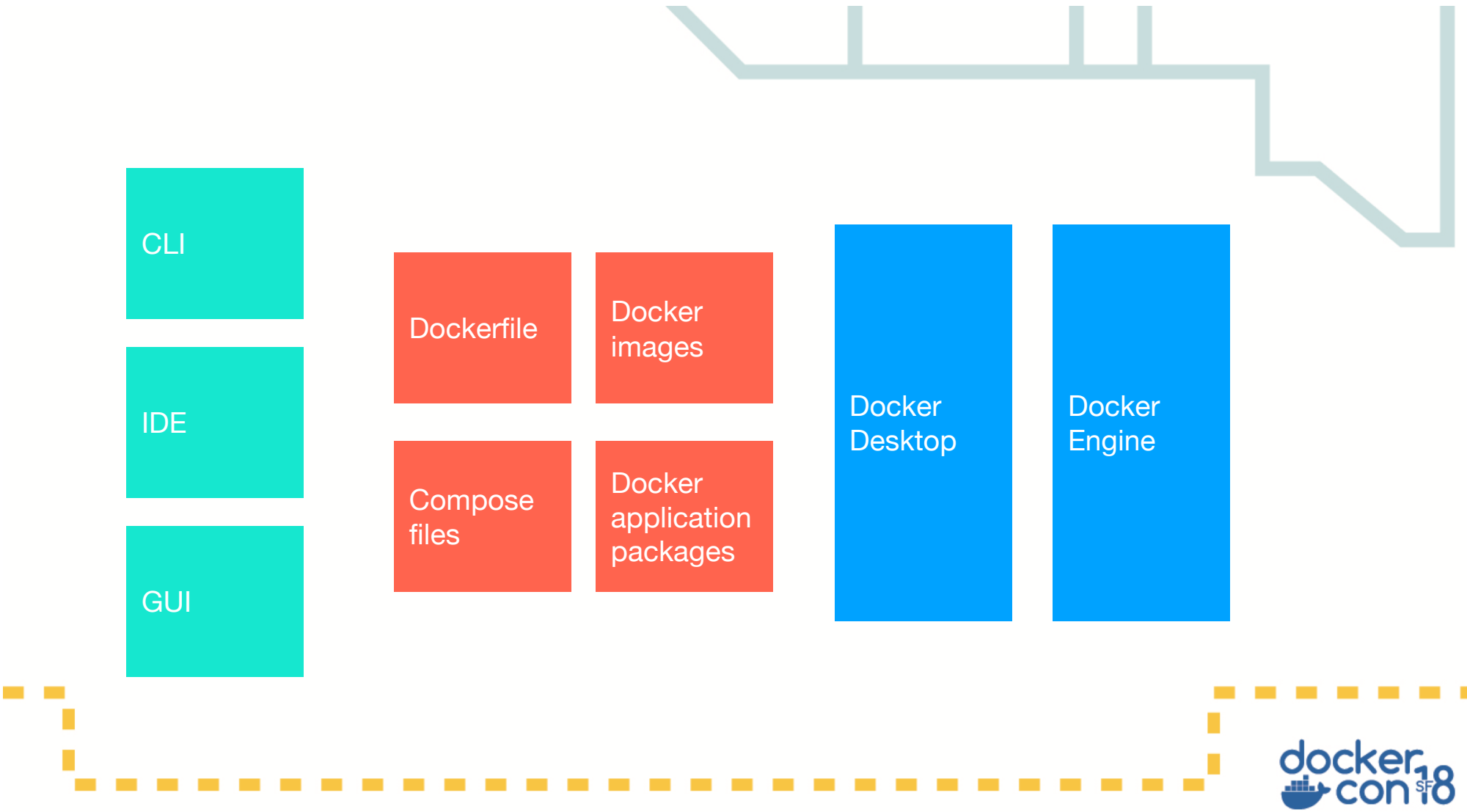
Coming Next

- **Collaboration and sharing**
  Share templates and services with your team
- **Deployment and CI integration**
  Integrate with CI/CD systems and deploy to EE
- **More features**
  Support for Volumes, Secrets and Networks, Content Trust and creation of application packages

Interested?

Sign up for the private beta at beta.docker.com

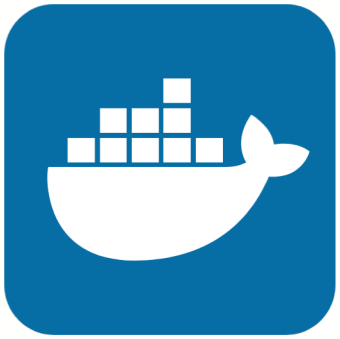

Wrapping Up
Docker that works for you
docker con SF 18


CLI
IDE
GUI
Dockerfile
Docker images
Compose files
Docker application packages
Docker Desktop
Docker Engine

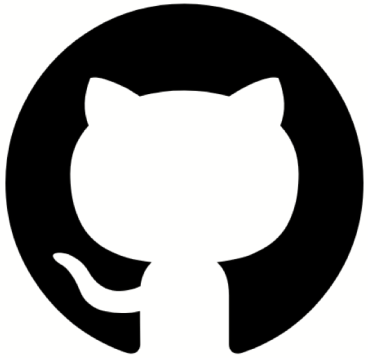
It's All About Developers

- **Docker Desktop**
  A fast and stable platform to develop applications on
- **Images and Applications**
  Better, and higher-level, tools to increase productivity
- **User interfaces**
  CLI, IDE and GUI based tools to cover all developers

Next steps

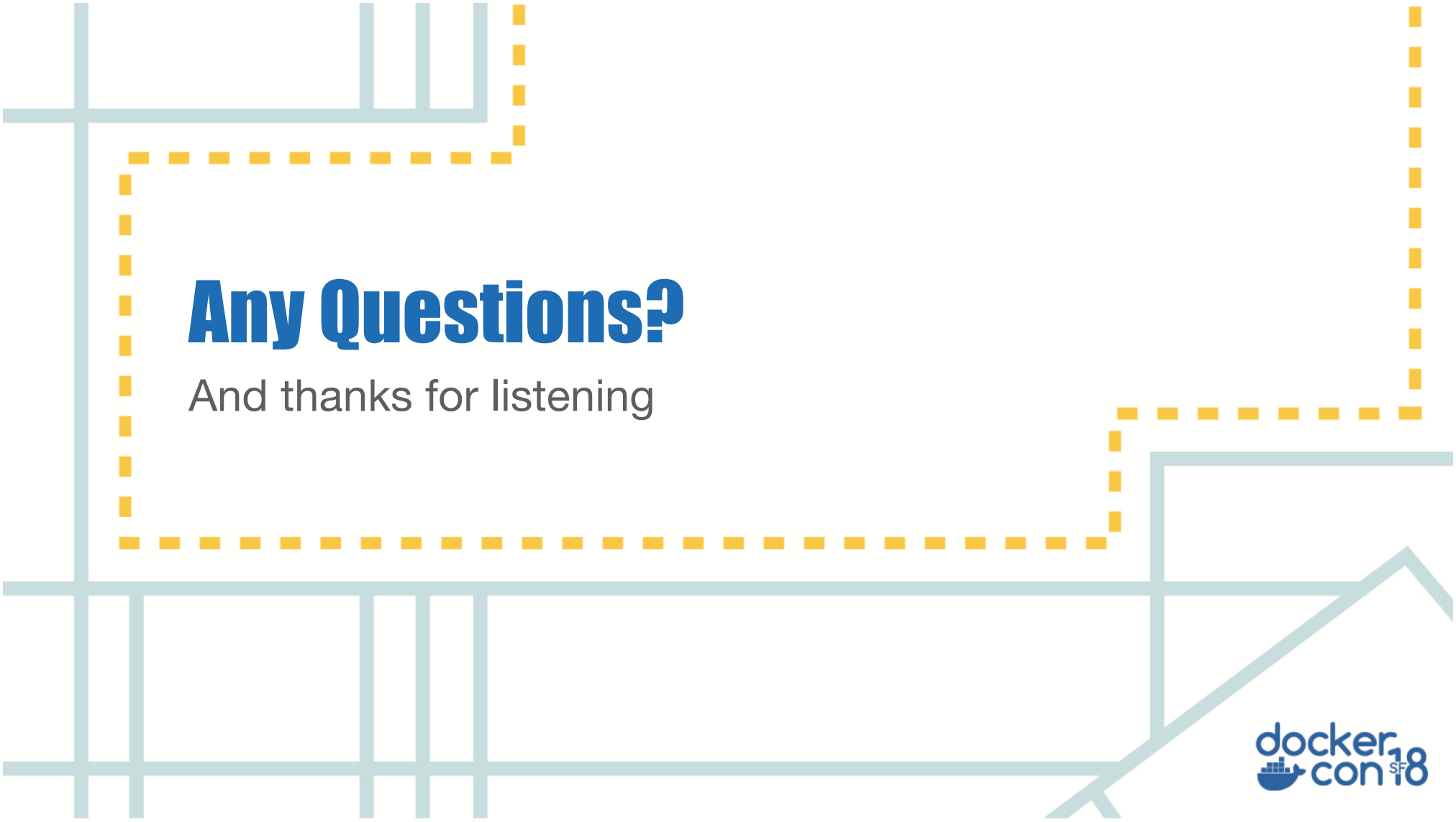


Download
Docker Desktop



Try out docker-
app

Join us on
Docker Community


Any Questions?
And thanks for listening
docker con SF 18