# Debugging Containers



Piotr Gaczkowski
IT Consultant

@doomhammer.info

#### Course Modules

Building Node Images

Configuring and Running Containers

**Debugging Containers** 

Interactive Debugging with IDEs

Running Multi-tier Applications with Docker Compose

#### Overview



Logging in containers

Logging in Express apps

Inspecting containers

Using Node.js debugger

Debugging Node.js apps from within the browser



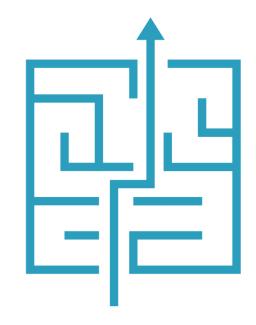
# Logging in Containers



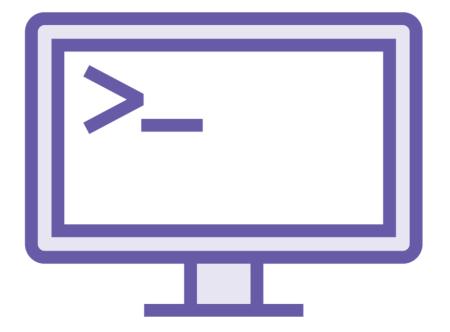
### Logging in Containers



Application containers run a single application



This approach simplifies logging



Standard output and standard error are logs



What's logged to the console is available to Docker



## Accessing Logs in Docker



docker logs command

- docker logs nginx

References container by its name or ID

Possible switches:

```
- --follow
```

- --since

- --tail (-n)

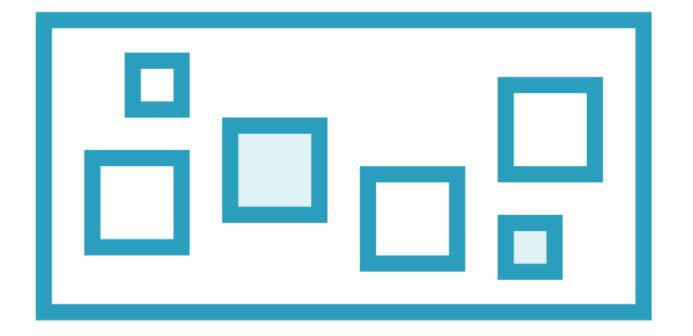
#### 

```
% docker run -d --name nginx nginx
b012f13b6e51fb7508c3d23d195b46c156dad6b110e7c1d8e7ab36aa3abe4484
% docker logs nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configurat
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.co
nf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2021/06/09 08:07:20 [notice] 1#1: using the "epoll" event method
2021/06/09 08:07:20 [notice] 1#1: nginx/1.21.0
2021/06/09 08:07:20 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)
2021/06/09 08:07:20 [notice] 1#1: OS: Linux 4.19.84-microsoft-standard
2021/06/09 08:07:20 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2021/06/09 08:07:20 [notice] 1#1: start worker processes
2021/06/09 08:07:20 [notice] 1#1: start worker process 32
2021/06/09 08:07:20 [notice] 1#1: start worker process 33
2021/06/09 08:07:20 [notice] 1#1: start worker process 34
2021/06/09 08:07:20 [notice] 1#1: start worker process 35
2021/06/09 08:07:20 [notice] 1#1: start worker process 36
2021/06/09 08:07:20 [notice] 1#1: start worker process 37
```



#### Other Uses of Container Logs

Manual inspection
Using Docker API to access logs
Log forwarding
Log drivers





#### Supplemental Pluralsight Course

Managing Advanced Kubernetes Logging and Tracing

Piotr Gaczkowski



console.log

Often the first choice for impromptu debugging

Single "log level"

No way to filter events

Not possible to configure

debug

Easy to turn on/off

Single "log level"

Possible to filter events:

```
DEBUG=express:* node index.js
DEBUG=* node index.js
```

Winston

**External module** 

Multiple log levels

**Multiple transports** 

Filtering and formatting

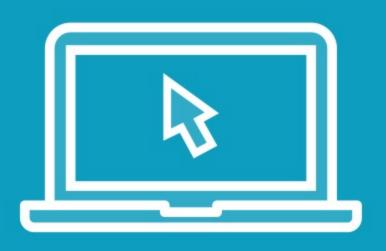
Configurable

### Winston Example

```
const { createLogger, format, transports } = require('winston')
const logger = createLogger({
 level: 'info',
  format: format.simple(),
 defaultMeta: { service: 'workout-gateway' },
 transports: [
   new transports.Console(),
});
function logRequest(req, res, next) {
   logger.info(req.url);
   next();
app.use(logRequest);
```



#### Demo



Add logging to the application

Run the application in a container

Get the logs from the container





docker inspect

docker exec



docker inspect

#### Works with:

- Containers
- Images
- Networks
- Volumes



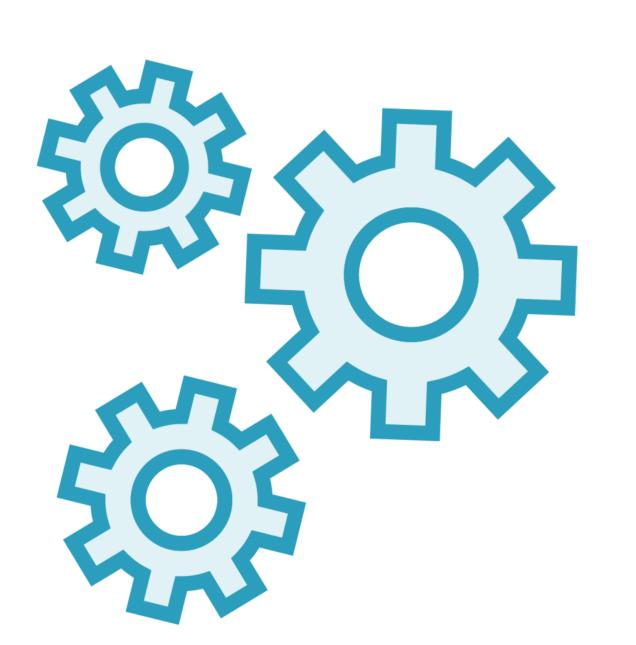
Host configuration

**Networking** 

**Volume mounts** 

**Container configuration** 

## Container Configuration



**Environment variables** 

**Entrypoint** 

**Working directory** 

Labels

User

Hostname

docker exec

# Executes another process in a running container

Share some options with docker run:

```
- -e/--env
- -i/--interactive
- -t/--tty
- -u/--user
- -w/--workdir
```

docker execuse use cases

Opening a shell in a container

Reloading the running process

Running auxiliary programs

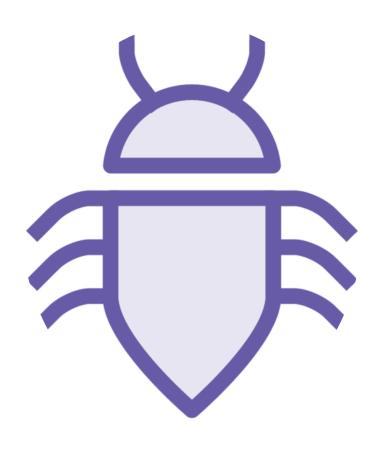
- Processing data
- Cleaning cache
- Checking open ports
- Periodic tasks



# Using Node.js Debugger



## Using Node.js Debugger







node --inspect



## Using Node.js Debugger

#### node inspect

Run an application with the debugger attached

Requires interactive terminal

Direct access to the debugger

Only suitable for terminal use

#### node --inspect

Run an application with an open port allowing debugger to attach

May be executed headless

```
docker run myapp:v2 node \
--inspect index.js
```

Requires remote debugger connection

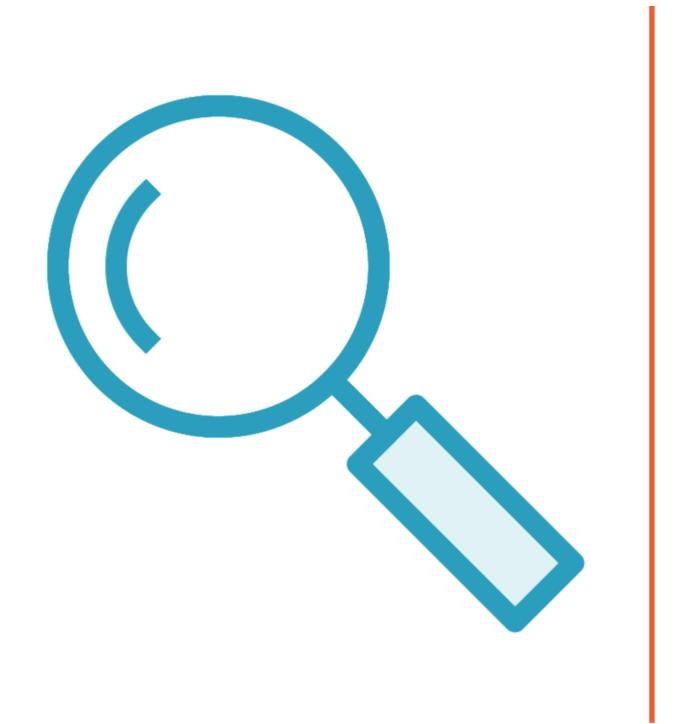
May be used from the browser or IDE



# Debugging in Browser



## Debugging in Browser

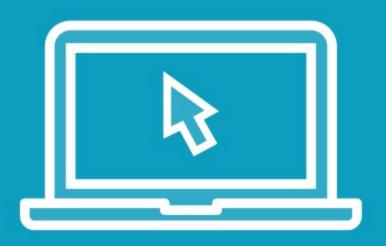


Uses node --inspect

#### Compatible with:

- Chrome DevTools
- Microsoft Edge

#### Demo



#### Debugging a Node.js Container

- Using docker inspect
- Running commands with docker exec
- Using the Node.js debugger

#### Summary



#### Different ways to troubleshoot containers:

- docker log
- docker inspect
- docker exec
- using the native debugger

Check networking whenever you application is misbehaving



#### References



Docker logs reference:

https://docs.docker.com/engine/reference/commandline/logs/



Winston homepage:

https://github.com/winstonjs/winston



**Debugging with Node.js:** 

https://nodejs.org/en/docs/guides/debugging-getting-started/

# Up Next: Interactive Debugging with IDEs

