Lesson 1

Jump Start with Nimbella

https://www.nimbella.com

Requirement for Certification

- complete the exercise
- complete a final online survey

Plan

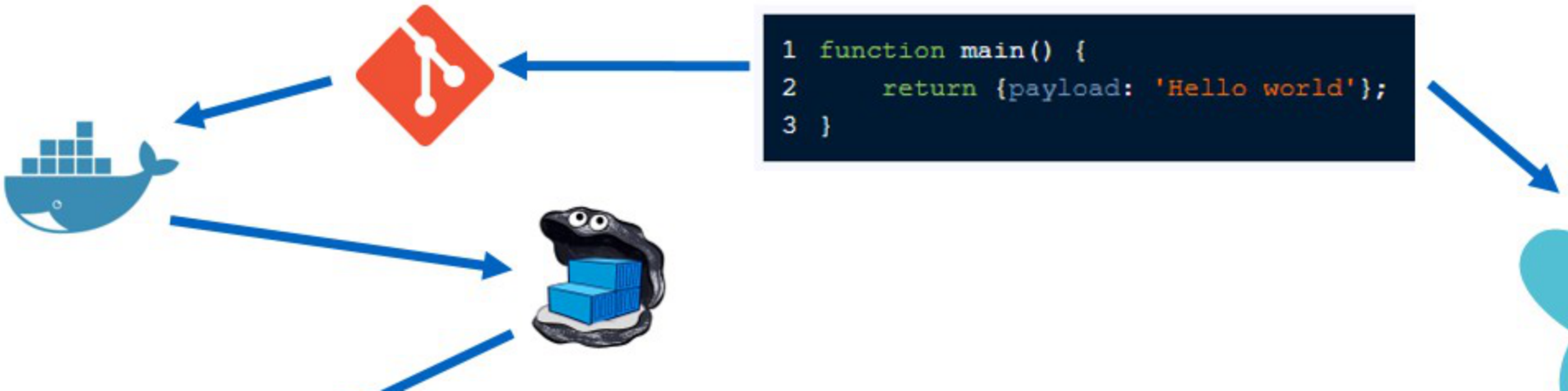
- Setup
  - signup with nimbella
  - installing using the `nim` cli
- Actions and Activations
  - creating an action with FAAS Wars
  - checking activation logs and results
- Triggers and Rules
  - a sample using slack
  - timed execution

Nimbella in a nutshell

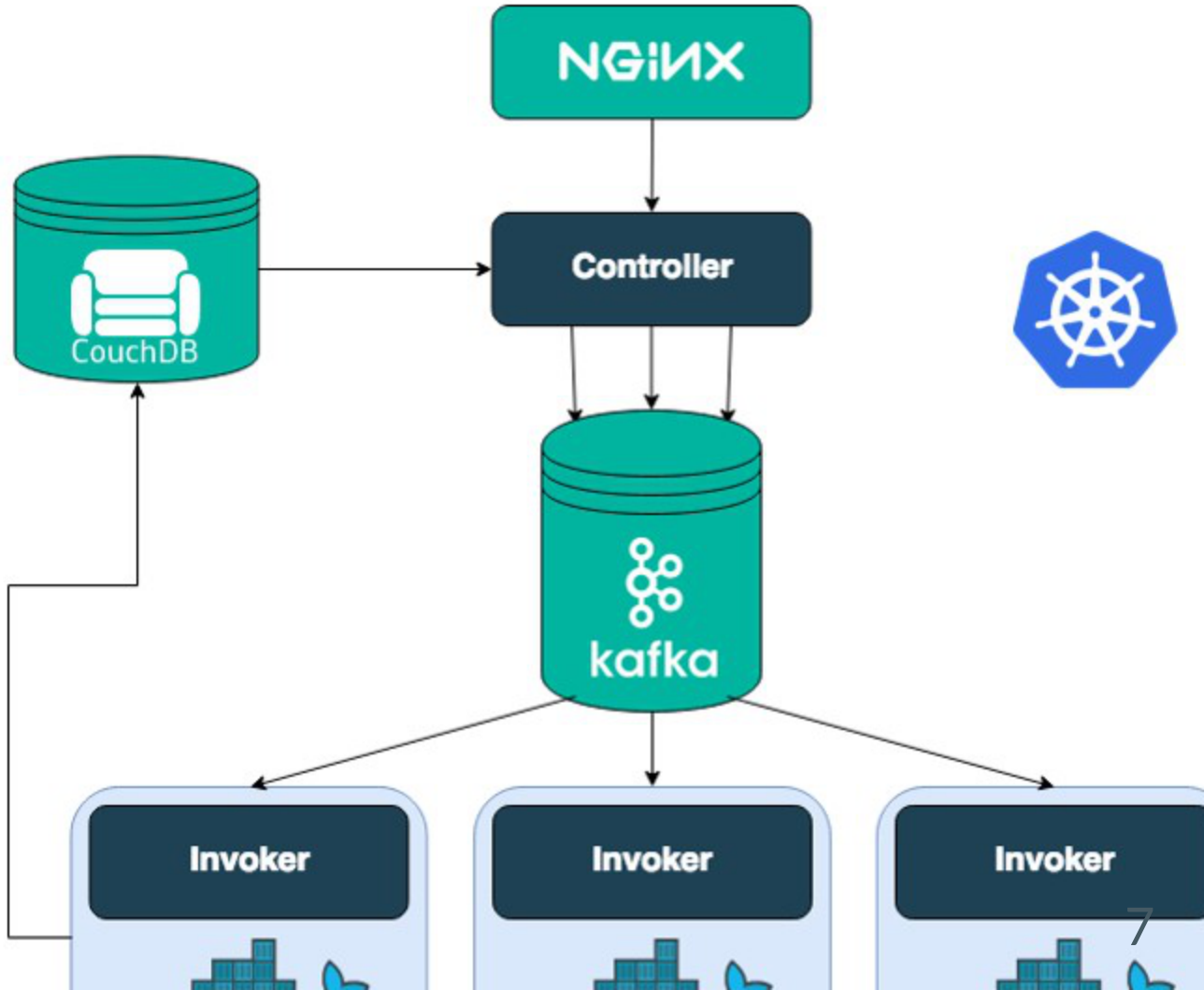
- Serveless Development Platform
- Cloud-Native made easy
- Microservice architecture
- Awesome developer experience
- Multi-cloud and "in your cloud"

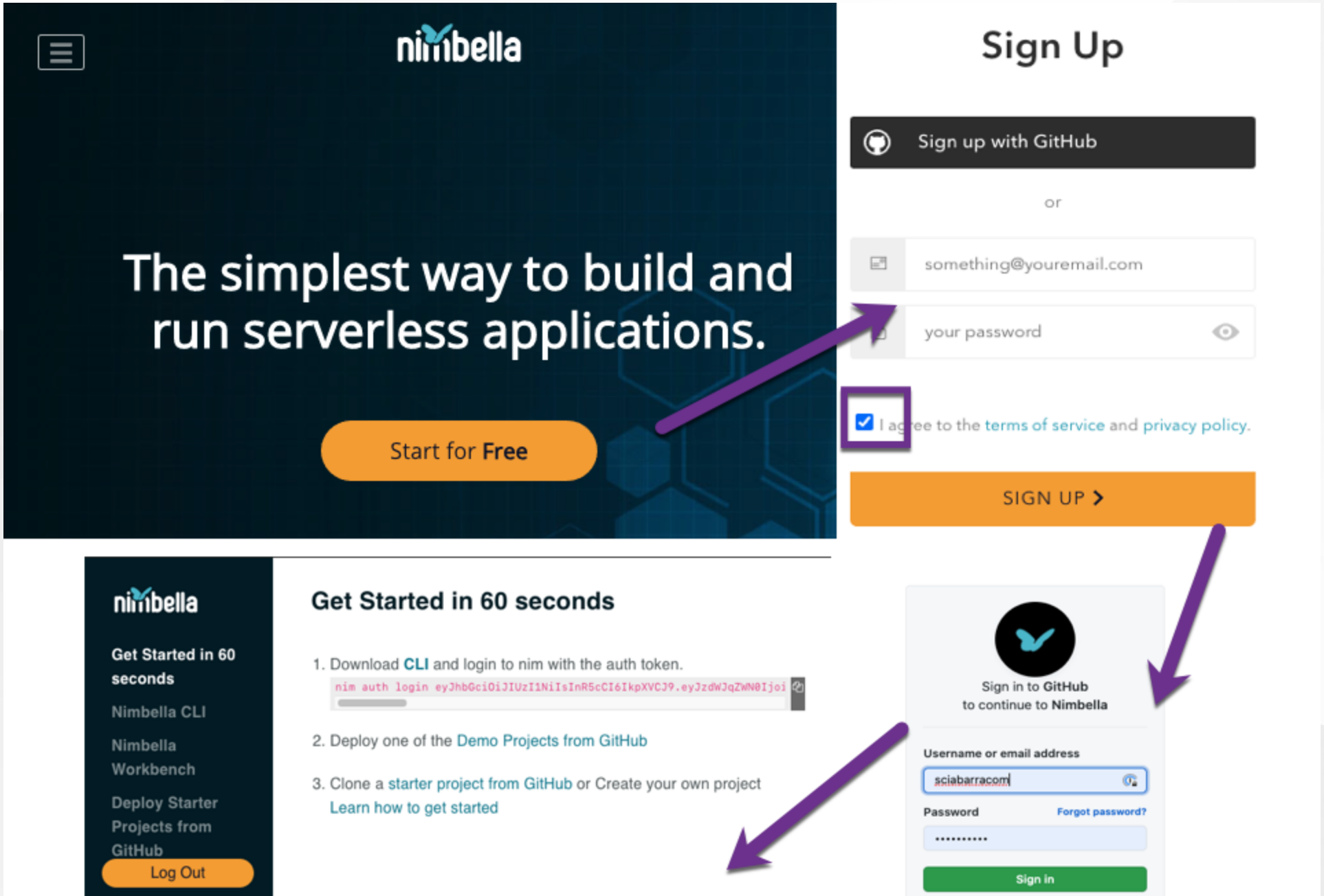
Nimbella vs Kubernetes vs Cloud

Kubernetes vs Nim



Nimbella Archit




nimbella
The simplest way to build and run serverless applications.
Start for Free
Sign Up
Sign up with GitHub
or
something@youremail.com
your password
I agree to the terms of service and privacy policy.
SIGN UP
nimbella
Get Started in 60 seconds
Nimbella CLI
Nimbella Workbench
Deploy Starter Projects from GitHub
Log Out
Get Started in 60 seconds
1. Download CLI and login to nim with the auth token.
nim auth login eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWJqZWN0Ijoi
2. Deploy one of the Demo Projects from GitHub
3. Clone a starter project from GitHub or Create your own project
Learn how to get started
Sign in to GitHub
to continue to Nimbella
Username or email address
sciabarracom
Password
Forgot password?
Sign in

Install nim cli

Nimbella CLI

The **Nimbella CLI** is called `nim`. It helps you organize and deploy your applications to the Nimbella cloud, to a secure domain that is unique to your projects. Select your platform below to download and install the CLI, then enter the following command to login and get stated.

Nimbella CLI

for Mac

Download

Nimbella CLI

for Linux

Download

Nimbella CLI

for Windows

Download

Platform Documentation (including CLI)

End User License Agreement

Login

- `nim auth login`
  open the browser and log into your github account
- `nim auth current`
  show your namespace
- `nim namespace get`
  show what you have in the namespace
- `nim namespace clean`
  cleaning your namespace

```
nim auth login
nim auth list
nim auth current
nim namespace get
nim namespace clean
```

Leaderboard

Red Fighter (Enemy)

andreati/shraduk

Cyan Fighter (You)

vverrast/codename

START THE BATTLE

1

LOGIN TO NIMBELLA

Red Fighter (Enemy)

matteope/Retry

Cyan Fighter (You)

Terminator

START THE BATTLE

EDIT MY FIGHTER

2

CREATE NEW FIGHTER

Jedi

SUBMIT TO FAAS WARS

JavaScript

3

```
function main(args){
    let actions = []
    switch(args.event) {
        case "idle":
            actions.push({"turn_turret_left": 45, "move_forwards": 50})
            actions.push({"turn_left": 45})
            break;
        case "wall-collide":
            actions.push({"move_opposide":10})
            actions.push({"turn_left":90})
            break
        case "hit":
            actions.push({"yell": "Ooops!"})
            break
        case "enemy-spot":
            actions.push({"yell": "Fire!", "shoot":true})
            break
        default:
            console.log(args)
    }
    return { "body": actions}
}
```

Inspecting Actions

- `nim action list`
  list actions
- `nim action get <name>`
  get informations about an action
- `nim action get <name> --url`
  get the public url of an action

```
# Inspecting the action
nim action list
nim action get Jedi
nim action get Jedi --url
```

Action invocation with nim

with `action invoke`:

- `nim action invoke <action-name> <parameters>`

`<parameters>`:

- `-p <name> <value> ...`
  can be repeated multiple times
- `-P <file>.json`
  you need a file in json format

```
# Invoking an action with parameters

nim action invoke Jedi -p event idle
nim action invoke Jedi -p event hit
nim action invoke Jedi -p event enemy-spot
nim action invoke Jedi -p event wall-collide

# invoking an action with json

echo '{ "event": "idle"}' >args.json
nim action invoke Jedi -P args.json
```

Action invocation with `curl`

only for web actions!

- `--web true`
  - web public it is the default with `nim`
  - **not** all the actions are web public

use url-encoded parameters

- `curl -X GET <url>?event=hit`
- `curl -X POST -d event=hit <url>`

```
# Using Curl  for web actions

URL=$(nim action get Jedi --url)
echo $URL

# use GET and url parameters

curl "$URL?event=hit"

## use POST and form data (url-encoded)

curl -X POST -d event=enemy-spot "$URL"
```

Updating an Action

- `nim action update <name> <file>`
  - works also if the action does not exists
  - some people only uses `update`

```
/*
nim action update Jedi jedi.js
nim action invoke Jedi
*/
function main(args) {
    console.log(args.event)
    return { body: [
        {"turn_turret_left": 15,
         "shoot": true}
    ]}
}
```

Checking Activations

- `nim activation list [--limit <n>]`
  list actions, you can limit them
- `nim activation logs [<id>]`
  show logs of an activation
- `nim activation result [<id>]`
  show logs of an activation

```
nim activation list
nim action invoke Jedi -p event idle
nim activation list --limit 3

nim activation logs
nim activation result
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1:

```
(base) $ wsk activation poll
Enter Ctrl-c to exit.
Polling for activation logs
```

```
(base) $ nim activation list
Datetime        Status   Kind      Version    Activation ID                    Start Wait Init Duration Entity
01/10 12:36:11  success  nodejs:10 0.0.6      c85ff06f059440ce9ff06f059450cee1 warm  157  0    2ms      Jedi
01/10 12:36:10  success  nodejs:10 0.0.6      a07ac392c08144c2bac392c081a4c297 warm  117  0    3ms      Jedi
01/10 12:36:10  success  nodejs:10 0.0.6      ef8aad1f5ac643ca8aad1f5ac6f3ca8e warm  112  0    2ms      Jedi
01/10 12:36:09  success  nodejs:10 0.0.6      b120ecdf4e044c50a0ecdf4e04bc507a warm  107  0    2ms      Jedi
01/10 12:36:09  success  nodejs:10 0.0.6      8df6bb9bd5db43afb6bb9bd5dba3af28 warm  4    0    2ms      Jedi
01/10 12:36:09  success  nodejs:10 0.0.6      413e21a0b2e740d1be21a0b2e740d110 warm  127  0    2ms      Jedi
01/10 12:36:08  success  nodejs:10 0.0.6      ece609ea66104d5fa609ea6610ad5f40 warm  118  0    2ms      Jedi
01/10 12:36:08  success  nodejs:10 0.0.6      00fe6f225ad74f7ebe6f225ad78f7e1f warm  108  0    2ms      Jedi
01/10 12:36:07  success  nodejs:10 0.0.6      6b55ac272c80407095ac272c800070ac warm  126  0    2ms      Jedi
01/10 12:36:07  success  nodejs:10 0.0.6      01aaac0df5e74e70aaac0df5e7ee70a5 warm  119  0    2ms      Jedi
01/10 12:36:06  success  nodejs:10 0.0.6      cd800a5132dd4b7f800a5132dd7b7f96 warm  187  0    2ms      Jedi
01/10 12:36:06  success  nodejs:10 0.0.6      bb386ea6d524443bb86ea6d524443b38 warm  405  0    2ms      Jedi
01/10 12:36:05  success  nodejs:10 0.0.6      444b50255c3140618b50255c3120617a warm  208  0    3ms      Jedi
01/10 12:36:05  success  nodejs:10 0.0.6      59cd6479ef6e4a258d6479ef6e1a255a cold  39   42   51ms     Jedi
01/10 12:32:25  success  nodejs:10 0.0.5      485d5c87bebc4b819d5c87bebcbb81e0 warm  125  0    2ms      Jedi
01/10 12:32:24  success  nodejs:10 0.0.5      525533b61f0040289533b61f0050288b warm  118  0    2ms      Jedi
01/10 12:32:24  success  nodejs:10 0.0.5      d24b03100dd141868b03100dd1c18600 warm  4    0    2ms      Jedi
01/10 12:32:24  success  nodejs:10 0.0.5      7d1fe404a8ff4d3c9fe404a8ff9d3c2c warm  110  0    2ms      Jedi
01/10 12:32:23  success  nodejs:10 0.0.5      4591162193734e8991162193732e8926 warm  5    0    2ms      Jedi
01/10 12:32:23  success  nodejs:10 0.0.5      e5dba7651cd7400b9ba7651cd7a00b86 warm  108  0    2ms      Jedi
01/10 12:32:23  success  nodejs:10 0.0.5      b1385677fab54f0db85677fab55f0db3 warm  4    0    2ms      Jedi
01/10 12:32:23  success  nodejs:10 0.0.5      2cf69bc8e8ea4815b69bc8e8ea8815b9 warm  124  0    2ms      Jedi
01/10 12:32:22  success  nodejs:10 0.0.5      290a1137d74c40668a1137d74c306614 warm  113  0    2ms      Jedi
01/10 12:32:22  success  nodejs:10 0.0.5      7212100811ce414e92100811ce414ef0 warm  106  0    2ms      Jedi
01/10 12:32:21  success  nodejs:10 0.0.5      0035e6c8d1c64363b5e6c8d1c6536376 warm  109  0    2ms      Jedi
01/10 12:32:21  success  nodejs:10 0.0.5      e1cf44c9776540138f44c97765301386 warm  112  0    2ms      Jedi
01/10 12:32:20  success  nodejs:10 0.0.5      8254bafcf845440e94bafcf845440ed0 warm  109  0    2ms      Jedi
```

Managing Packages

package = "collection of actions"

- `nim package create greetings`
  create a package
- `nim action create greetings/hello hello.js`
  create an action in the package
- `nim action remove greetings -r`
  remove package and all its actions

```
# create package with 2 actions
nim package list
nim action list
nim package create greetings
nim action create greetings/hello hello.js
nim action create greetings/hi hi.js
# check and clean
nim package list
nim action list
nim package delete greetings
nim package delete greetings -r
nim package list
nim action list
```

Package variables

- `nim package update -p name Mike`
- available to all actions in package

Action variables

- `nim action update -p name Mike`
- useful to share configurations
- action variables overrides package variables

```
# create package without variables
nim package create greetings
cat hello.js
nim action create greetings/hello hello.js
cat hi.js
nim action create greetings/hi hi.js

# no variables, default
nim action invoke greetings/hello
nim action invoke greetings/hi
```

```
# override package variable
nim package update greetings -p name Mike
nim action invoke greetings/hello
nim action invoke greetings/hi

# override action variable
nim action update greetings/hi -p name Michele
nim action invoke greetings/hello
nim action invoke greetings/hi
```

Shared Packages in `whisk-system`

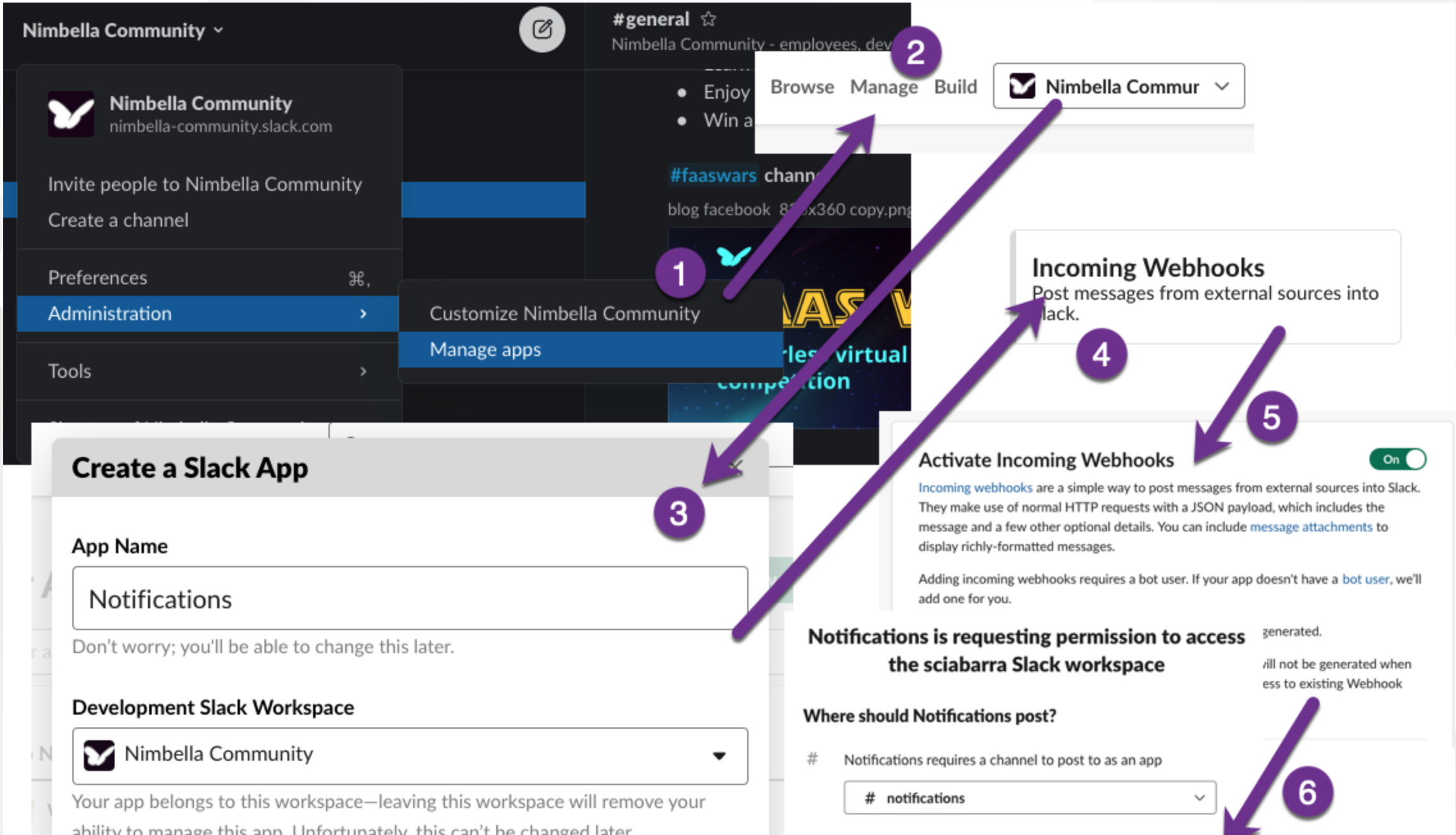
- you can share your package with others:
  `nim action create <package> --shared=yes`
- shared (system) packages:
  `nim packages list /whisk-system`

Datetime	Access	Kind	Version	Packages
06/18 22:16:37	public	package	0.0.11	alarms
02/21 13:35:18	public	package	0.0.2	utils

```
nim package list /whisk-system
nim action list /whisk-system/alarms
nim action get /whisk-system/alarms/interval
```

Example: a "slack" notification action

1. Creating a slack URL
2. Passing the URL as package variable
3. Writing messages in the url with an action
4. Profit!


Nimbella Community
#general
Nimbella Community - employees, dev
Nimbella Community
nimbella-community.slack.com
Invite people to Nimbella Community
Create a channel
Preferences ⌘,
Administration
Tools
Customize Nimbella Community
Manage apps
Browse Manage Build
Nimbella Commur
#faaswars chann
blog facebook
1
2
3
4
5
6
Incoming Webhooks
Post messages from external sources into lack.
Create a Slack App
App Name
Notifications
Don't worry; you'll be able to change this later.
Development Slack Workspace
Nimbella Community
Your app belongs to this workspace—leaving this workspace will remove your
Activate Incoming Webhooks
On
Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details. You can include message attachments to display richly-formatted messages.
Adding incoming webhooks requires a bot user. If your app doesn't have a bot user, we'll add one for you.
Notifications is requesting permission to access the sciabarra Slack workspace
generated.
ill not be generated when
ess to existing Webhook
Where should Notifications post?
Notifications requires a channel to post to as an app
notifications

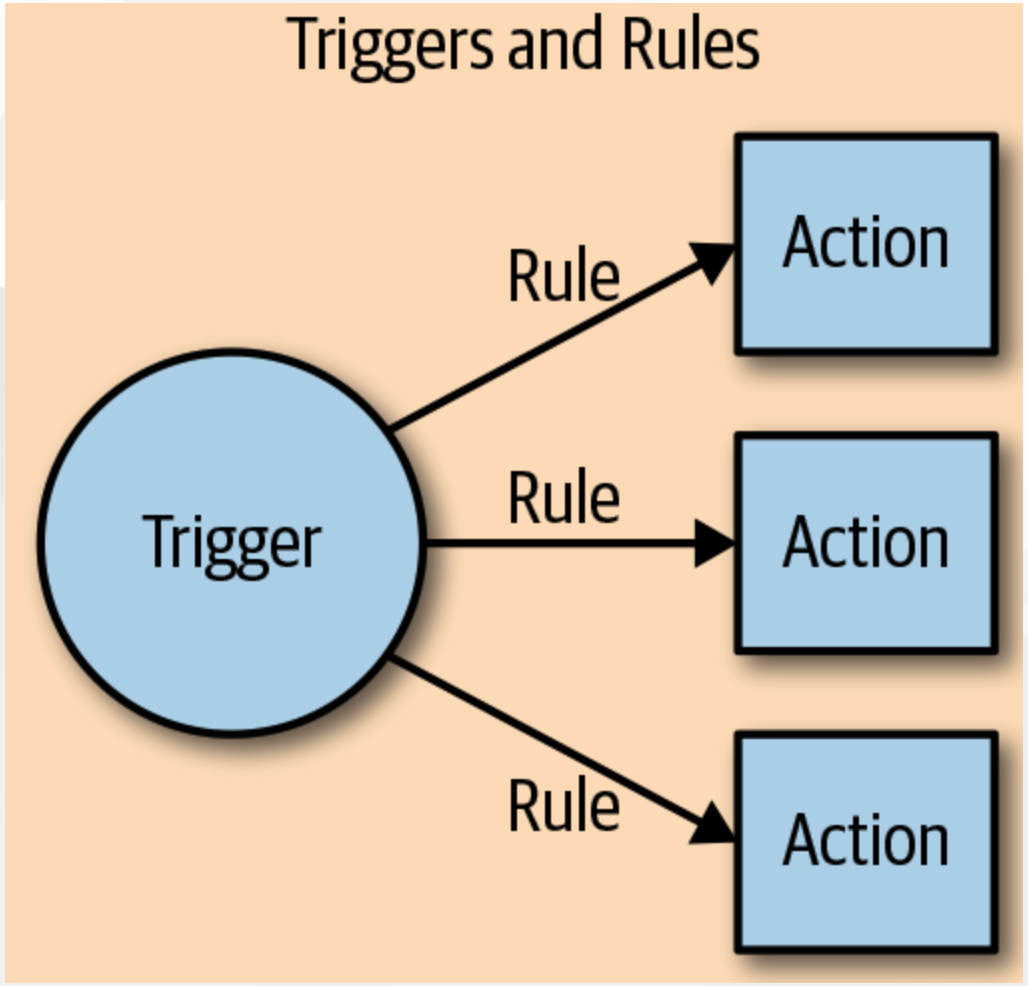
```
# opening slack
source $HOME/.ssh/secret.sh
curl -X POST -d '{"text": "Hello"}' $NOTIFICATIONS
curl -X POST -d '{"text": "How are you"}' $NOTIFICATIONS
```

```
// notify.js
const axios = require('axios').default;

function main(args) {
    return axios.post(args.notifications, {
        text: args.text
    }).then(r => {
        return {
            "body": r.data
        }
    })
}
```

```
# Notifications
nim package update slack -p notifications $NOTIFICATIONS
nim action update slack/notify notify.js
nim action invoke slack/notify -p text hello
nim action invoke slack/notify -p text hi
```

Triggers and Rules



Creating a trigger

- `nim trigger create <name>`
  create a trigger with the given `<name>`
- `nim trigger fire <name> <parameters>`
  fire a trigger with parameters
- `<parameters>`:
  - `-p <name> <value>...`
  - `-P <file>.json`

Creating and enabling rules

- `nim rule create <name> <trigger> <action>`
  - create a rule `<name>`
  - ...to invoke the `<action>`
  - ... when firing the `<trigger>`
- `nim rule enable <name>`
  `nim rule disable <name>`
  enable or disable rules

```
// echo.js
function main(args) {
    console.log(args)
    return args
}
```

```
# inspecting trigger invocation
nim trigger create echoer
nim action create echo echo.js
nim rule create echoer-echo echoer echo
nim trigger fire echoer -p hello world
nim activation logs
nim activation result
```

```
// notify2.js
// prefixed message
// parameters from trigger
const axios = require('axios').default;

function main(args) {
    let prefix = args.prefix || ""
    let text = args.parameters[0].value
    return axios.post(args.notifications, {
        text: prefix + text
    }).then(r => {
        return {
            "body": r.data
        }
    })
}
```

```
nim trigger create slacker
nim action update slack/first notify2.js -p prefix '[first] '
nim rule create slacker-first slacker slack/first
nim rule enable slacker-first
nim trigger fire slacker -p text from-trigger-1
```

```
nim action update slack/second notify2.js -p prefix '[second] '
nim rule create slacker-second slacker slack/second
nim rule enable slacker-second
nim trigger fire slacker -p text from-trigger-2
nim rule disable slacker-first
nim trigger fire slacker -p text from-trigger-3
```

Feed

- a feed is a sorce of events for a trigger
  - it will fire when an event occurs

Alarms

- predefined Nimbella feed
- generate periodically trigger invocations
- ```
  nim trigger create every-minute \
  --feed /whisk-system/alarms/interval -p minutes 1
  ```
  ```

```
# Inspecting packages
nim package list /whisk-system/
nim action list /whisk-system/alarms
nim action get /whisk-system/alarms/interval
```

```
const axios = require('axios').default;

function main(args) {
    let text = new Date().toISOString()
    return axios.post(args.notifications, {
        text: text
    }).then(r => {
        return {
            "body": r.data
        }
    })
}
```

```
nim trigger create ticker --feed /whisk-system/alarms/interval -p minutes 1
nim action update slack/tick tick.js
nim rule create ticker-tick ticker slack/tick
nim rule enable ticker-tick
```

Exercise for certification

- create a web site monitor
  - checks if a site is up and running
  - notifies in slack if something is wrong
  - it is executed every minute