dreamforce 20 Years

Design Patterns for Salesforce CI/CD

Pablo Gonzalez, Business Engineering Architect, Salto
https://www.linkedin.com/in/pablis/

Pablo Gonzalez

Business Engineering Architect @ Salto.io

Agenda

- Mandatory history on CI/CD
- Patterns for creating a Salesforce deployment pipeline
- CI with GitHub actions
- Full demo

Continuous Integration (CI)

The problems

CI aims to solve

Combining the work of multiple developers is hard

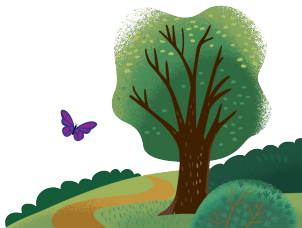
Developers have adopted branches to work in isolated environments

Branches diverge from each other

The more branches you have, the harder it is to merge them together

Complicated merges lead to code freezes

Expensive and unpredictable process



Continuous Integration

defined

- Continuous Integration is a software development practice where members of a team integrate their work frequently into a common branch of a version control repository
- Each integration is verified by an automated build to detect integration errors as quickly as possible.

Challenges of Continuous Integration

Tasks need to be broken down into small chunks

Paradigm shift for most developers

Need for reliable automated regression testing

Hard to implement in Salesforce due to governor limits, inconsistent testing frameworks, etc.

Continuous Delivery (CD)

Continuous Delivery is the ability to get changes of all types into production, safely and quickly in a sustainable way.

Jez Humble, Author of continuousdelivery.com

Continuous Delivery

principles

Build quality in

Fix bugs as soon as they are found and ideally before committing them to version control

Avoid reliance on manual testing

Work in small batches

Get feedback as quickly as possible

Automate everything

Let people focus on higher value activities

Continuous Improvement

CI/CD is not the goal

Don’t be satisfied with the status quo

Salesforce Deployment Pipeline

An automated process that runs your salesforce metadata through a series of steps such as quality checks, tests and deployments. This process fires in response to events in your version control repository.

Each successful step increases confidence in our implementation.

Patterns, not mandates

Do whatever works for you

Choosing a sandbox strategy

Step 1

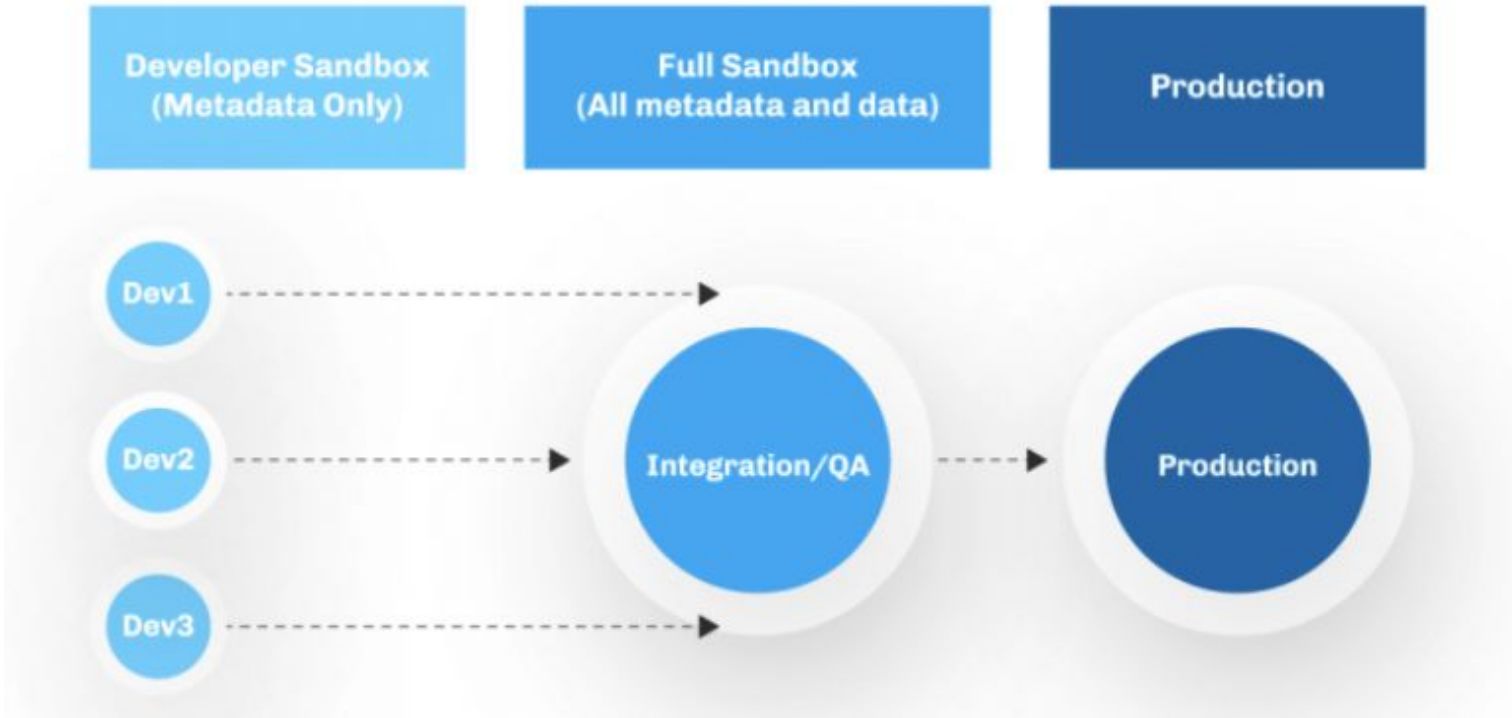
Sandbox Strategy

Things to consider

- Should every developer have their own sandbox?
- Which type of sandbox will you use for different types of testing?
- How are sandbox populated with fake data?
- How easily can you refresh your full sandbox (post-refresh activities)

Sandbox Strategy

Typical strategy



Choosing which metadata to track

Step 2

Tracking metadata in git

Things to consider

- Should we track only code-based metadata?
- What about changes that are made directly in production (email templates, deactivation a flow, etc)
- Do we know the Salesforce metadata API well enough to track everything?
- Which metadata, if versioned, will make our releases and apps **better**?

Choosing a branching strategy

Step 3

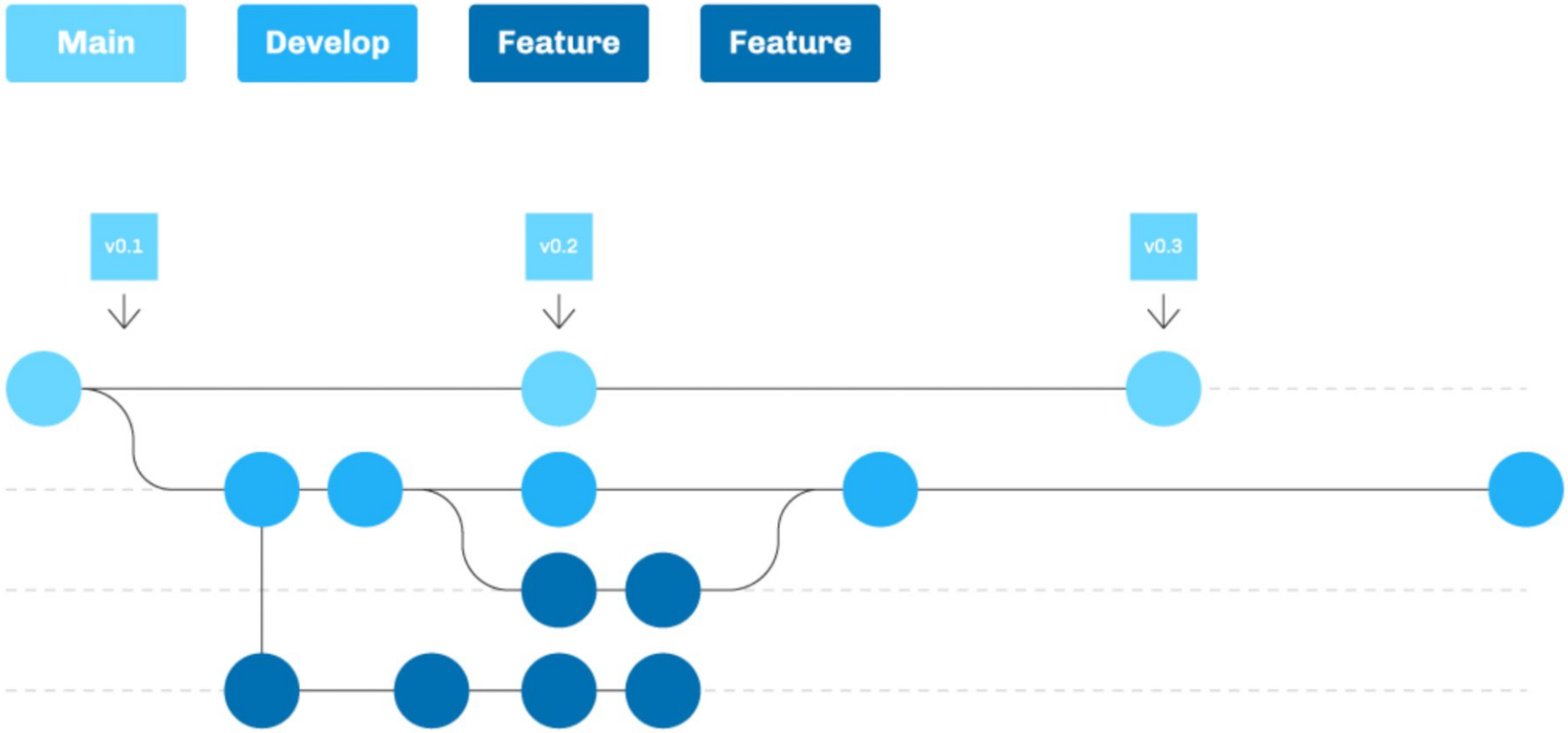
Branching strategy

Things to consider

- Do we create one branch per Salesforce org?
- Is there a version of trunk-based development that would fit our needs?
- How do we deploy to production? from which branch and when?
- Aim to have **short-lived** branches

Branching strategy

Gitflow



Choosing a CI server

Step 4

A CI server creates virtual machine

that can do the following

Check out your sfdx project

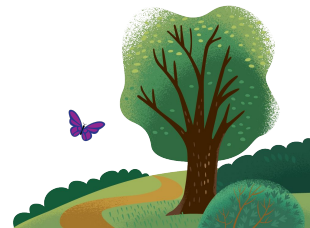
It loads your tracked sfdx project into its file system

Respond to events in version control

Will listen to push, pull requests, and other events

Execute commands on a terminal

sfdx commands, bash, node.js, etc.



CI servers

Options

- GitHub Actions
- Azure DevOps
- Bitbucket Pipelines
- GitLab CI
- Circle CI
- Jenkins

Deciding what to automate and when

Step 5

Actions

to automate

Deployment

Automate the deployment of metadata to the next org in the pipeline

Run tests specified by the developer

Give the developer the freedom to choose which tests should be run

Scan the code

Use PMD to scan the apex code

Delta deployment

Deploy only the metadata that has been created/updated since the last commit

Events

to trigger automation

Pull request is open

- Figure out which metadata has changed
- PMD
- Check-only deployment of delta to INT org
- Run tests specified by developer

Pull request is approved and merged

- Full deployment to UAT
- Run all tests

Development branch is merged into master

- Full deployment to production
- Run all tests

Finally! a demo!

Workflow with GitHub actions

Authenticate to target org

how it's done

```
136        # The URL is stored in the Github Secret named SFDX_INTEGRATION_URL
137        # so here we store the URL into a text file
138        - name: "Populate auth file with SFDX_URL secret of integration org"
139          shell: bash
140          run: |
141            echo ${{ secrets.SFDX_INTEGRATION_URL}} > ./SFDX_INTEGRATION_URL.txt
142
143        # Authenticate to org using the URL stored in the text file
144        - name: "Authenticate to Integration Org"
145          run: sfdx auth:sfdxurl:store -f ./SFDX_INTEGRATION_URL.txt -s -a integration
146
```

Run apex tests specified in pull request

how it's done (part 1)

pgonzaleznetwork commented 18 days ago

Description

Please include a summary of the change and what has changed.

Jira Ticket

CRM-XXX

Apex Tests to Run

Apex::[GitClassTest]::Apex

Run apex tests specified in pull request

how it's done (part 2)

```
85      run: |
86
87        FILE=./parsePR.js
88        if test -f "$FILE"; then
89            echo $PR_BODY > ./pr_body.txt
90            node ./parsePR.js
91            TESTS=$(cat testsToRun.txt)
92            echo "APEX_TESTS=$TESTS" >> $GITHUB_ENV
93        else
94            TESTS=all
95            echo "APEX_TESTS=$TESTS" >> $GITHUB_ENV
96        fi
97
```

Run apex tests specified in pull request

how it's done (part 3)

```
10      const lines = readline.createInterface({
11          input: fs.createReadStream(__dirname+'/pr_body.txt'),
12          crlfDelay: Infinity
13      });
14
15      for await (const line of lines) {
16
17          let upperLine = line.toUpperCase();
18
19          //special delimeter for apex tests
20          if(upperLine.includes('APEX::[') && upperLine.includes(']::APEX')){
21
22              let tests = line.substring(8,line.length-7);
23              await fs.promises.writeFile(testsFile,tests);
24              await fs.promises.appendFile(testsFile,'\n');
25          }
26      }
27  }
```

Generate delta deployment

how it's done (sfdx-git-delta)

```
147        # We use SFDX Git Delta to create a directory with only the metadata that has changed.
148        # this allows us to deploy only those changes, as opposed to deploying the entire branch.
149        # This helps reducing deployment times
150        - name: "Create delta packages for new, modified or deleted metadata"
151          run: |
152            mkdir changed-sources
153            sfdx sgd:source:delta --to "HEAD" --from "HEAD^" --output changed-sources/ --generate-delta --source force-app/
154
```

Extra

topics I didn’t cover

Docker

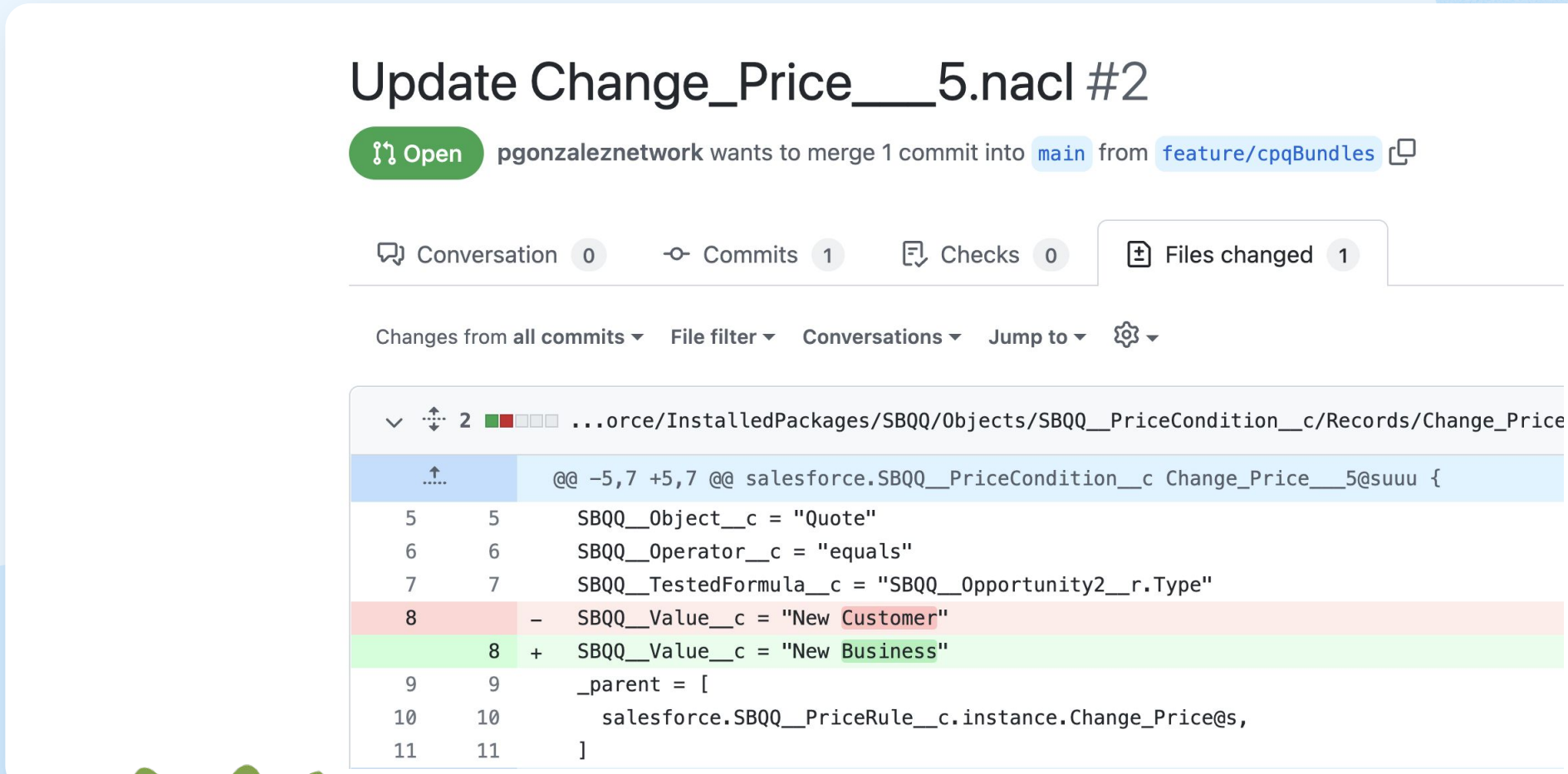
containers/images for CI servers

Dockerfile

```
1   FROM heroku/heroku:18
2
3   ENV DEBIAN_FRONTEND=noninteractive
4   ARG SALESFORCE_CLI_VERSION=latest-rc
5   ARG SF_CLI_VERSION=latest-rc
6
7   RUN echo 'a0f23911d5d9c371e95ad19e4e538d19bffc0965700f187840eb39a91b0c3fb0  ./nodejs.tar.gz' > node-file-lock.sha \
8     && curl -s -o nodejs.tar.gz https://nodejs.org/dist/v16.13.2/node-v16.13.2-linux-x64.tar.gz \
9     && shasum --check node-file-lock.sha
10  RUN mkdir /usr/local/lib/nodejs \
11    && tar xf nodejs.tar.gz -C /usr/local/lib/nodejs/ --strip-components 1 \
12    && rm nodejs.tar.gz node-file-lock.sha
13
14  ENV PATH=/usr/local/lib/nodejs/bin:$PATH
15  RUN npm install --global sfdx-cli@${SALESFORCE_CLI_VERSION} --ignore-scripts
16  RUN npm install --global @salesforce/cli@${SF_CLI_VERSION}
17
```

CI/CD for configuration data (CPQ, etc.)

NaCl (open source)

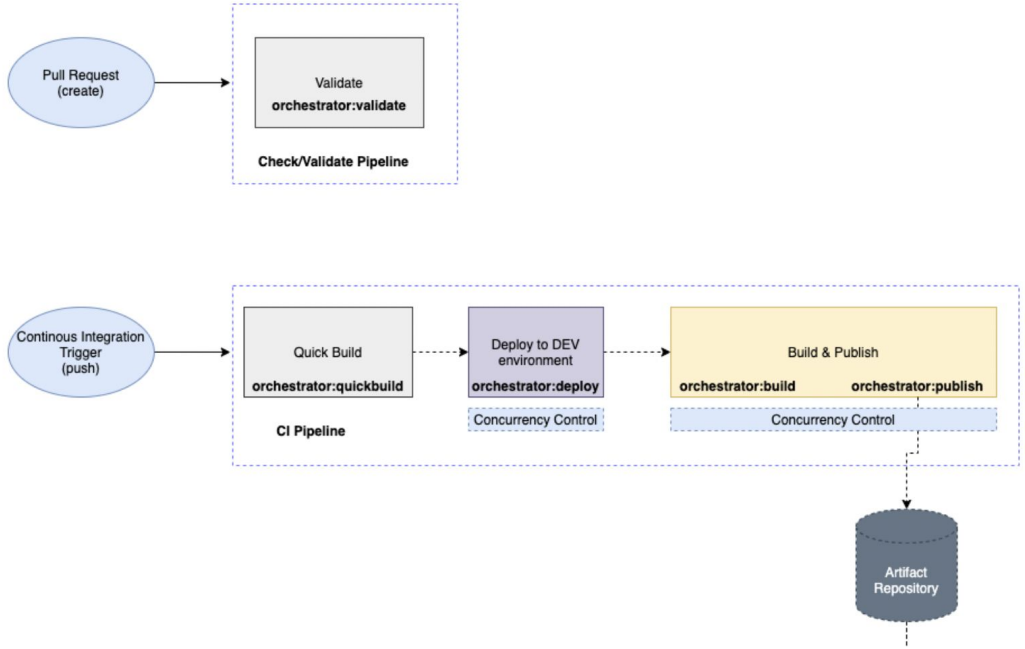


DX@Scale

Modular development

A Typical CI/CD Pipeline

Let's look at a typical CI/CD pipeline for a package-based development in a program that has multiple environments. For brevity, validation before integration is not discussed



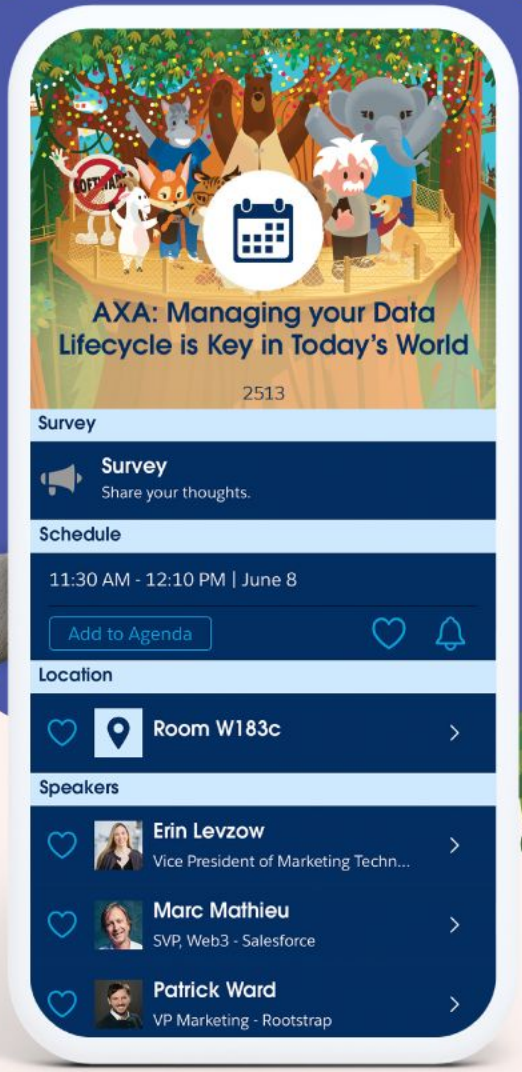
HappySoup.io/cicd

Summary and link to free auto-CI app

dreamforce 20 Years

Share your feedback.

Provide your feedback on this session in the Salesforce Events mobile app and help make our content even better.



#DF22


salesforce
Thank you