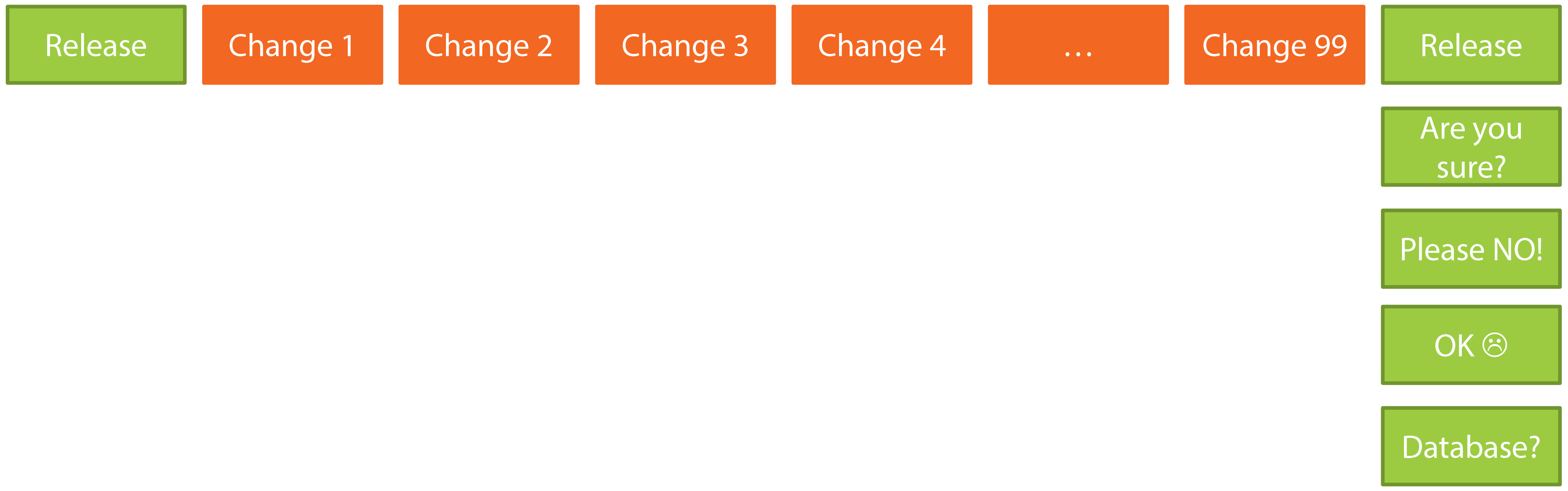
Rethinking How We Develop

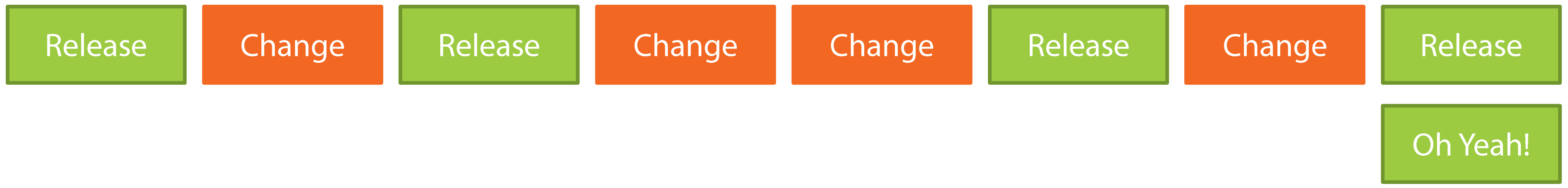
Wes Higbee

@g0t4 | www.weshigbee.com

Traditional Development



Delivering Frequently



Why Delivery Frequently?

- What else is left to be confident?
- Less risk
- Value sooner
- Avoid branching
- Confidence
- Changes don’t pile up
- Faster releases
- Happier users (potentially)

How, Roughly

- Prioritize business value (employees, customers, organization)
- Prioritize deliverability (value)
- Fix problems fast
- One thing at a time, as a team
  - Avoid branches
  - Use toggles
- Safe database changes
  - Tested, confidence
- Align incentives of everyone involved

Developer Incentives

- DB history with code
- Confidence
- Own island to experiment
- No more interruptions
- Refactor bad DB design
- Stop cramming things where they don't belong
- No more reverse engineering / schema compare at last minute
- Can write tests with DB easily

DBA Incentives

- Review changes as they happen
- Small changes
- Not rushed
- No throwing SQL over the wall
- Process to test their changes
- Teach others
- Automated tool, less change control sign off TPS reports

Ops Incentives

- Automation
- Process they can rely upon
- Less fires
- Smaller maintenance windows
- Process they can help create, believe in

Tester Incentives

- Recreate past DB versions
- Automated testing with DB
- Automated exploratory testing environments
- Less delay

Leaders Incentives

- Automated, accurate change management
- Less hovering about TPS reports
- More time spent creating value
- Value sooner
- Less risk
- Less problems at release
- Agile response to change in priorities

Customer Incentives

- Value sooner
- Less bugs
- Faster turn around for requests
- More value: less time spent on process

Code Based Migrations

```sql
CREATE TABLE Contexts
(
    Id INT PRIMARY KEY NOT NULL IDENTITY,
    Name NVARCHAR(255) NOT NULL
);
```

```csharp
[Migration(20090906205342)]
public class AddGTDTables : Migration
{
        public override void Up()
        {
                Create.Table("Contexts")
                        .WithIdColumn()
                        .WithColumn("Name").AsString().NotNullable();
```

```
public class CreateViewsMigration : Migration
{
     public override void Up()
     {
        IfDatabase("oracle").Execute.Script("CreateViewsOracleMigrationUp.
        IfDatabase("sqlserver").Execute.Script("CreateViewsSqlServerMigrat
     }
```

```sql
--changeset nvoxland:1
create table person (
  id int not null primary key,
  firstname varchar(80),
  lastname varchar(80) not null,
  state varchar(2)
);
```

```xml
<?xml version="1.0" encoding="UTF-8"?>

<databaseChangeLog
        xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
        xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog ht
tp://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd
        http://www.liquibase.org/xml/ns/dbchangelog-ext http://www.liquiba
se.org/xml/ns/dbchangelog/dbchangelog-ext.xsd">

    <preConditions>
        <runningAs username="liquibase"/>
    </preConditions>

    <changeSet id="1" author="nvoxland">
        <createTable tableName="person">
            <column name="id" type="int" autoIncrement="true">
                <constraints primaryKey="true" nullable="false"/>
            </column>
            <column name="firstname" type="varchar(50)"/>
            <column name="lastname" type="varchar(50)">
                <constraints nullable="false"/>
            </column>
            <column name="state" type="char(2)"/>
        </createTable>
    </changeSet>

    <changeSet id="2" author="nvoxland">
```

```javascript
exports.up = function (db, callback) {
  db.createTable('pets', {
    id: { type: 'int', primaryKey: true },
    name: 'string'
  }, callback);
};

exports.down = function (db, callback) {
  db.dropTable('pets', callback);
};
```

```
exports.up = function (db, callback) {
  db.createTable('pets', {
    id: { type: 'int', primaryKey: true },
    name: 'string'
  }, createOwners);

  function createOwners(err) {
    if (err) { callback(err); return; }
    db.createTable('owners', {
      id: { type: 'int', primaryKey: true },
      name: 'string'
    }, callback);
  }
};

exports.down = function (db, callback) {
  db.dropTable('pets', function(err) {
    if (err) { callback(err); return; }
    db.dropTable('owners', callback);
  });
};
```

```javascript
var async = require('async');

exports.up = function (db, callback) {
  async.series([
    db.createTable.bind(db, 'pets', {
      id: { type: 'int', primaryKey: true },
      name: 'string'
    }),
    db.createTable.bind(db, 'owners', {
      id: { type: 'int', primaryKey: true },
      name: 'string'
    });
  ], callback);
};

exports.down = function (db, callback) {
  async.series([
    db.dropTable.bind(db, 'pets'),
    db.dropTable.bind(db, 'owners')
  ], callback);
};
```

Safe Change

NOW

```
ALTER TABLE employees
ADD name VARCHAR(80) NULL
GO

UPDATE employees set name = CONCAT(first_name, ' ', last_name)
GO

ALTER TABLE employees DROP column last_name
ALTER TABLE employees DROP column first_name
GO

ALTER VIEW employee_positions AS
SELECT employees.id AS employee_id,
        name,
        title
FROM employees
        LEFT JOIN titles on employees.title_id = titles.id
```

Safe Change

NOW

```
ALTER TABLE employees
ADD name VARCHAR(80) NULL
GO

UPDATE employees set name = CONCAT(first_name, ' ', last_name)
GO




ALTER VIEW employee_positions AS
SELECT employees.id AS employee_id,
        name,
        title
FROM employees
        LEFT JOIN titles on employees.title_id = titles.id
```

LATER

```
ALTER TABLE employees DROP column last_name
ALTER TABLE employees DROP column first_name
GO
```

Safe Change

NOW

```
ALTER TABLE employees
ADD name VARCHAR(80) NULL
GO

UPDATE employees set name = CONCAT(first_name, ' ', last_name)
GO
```

CREATE TRIGGERS

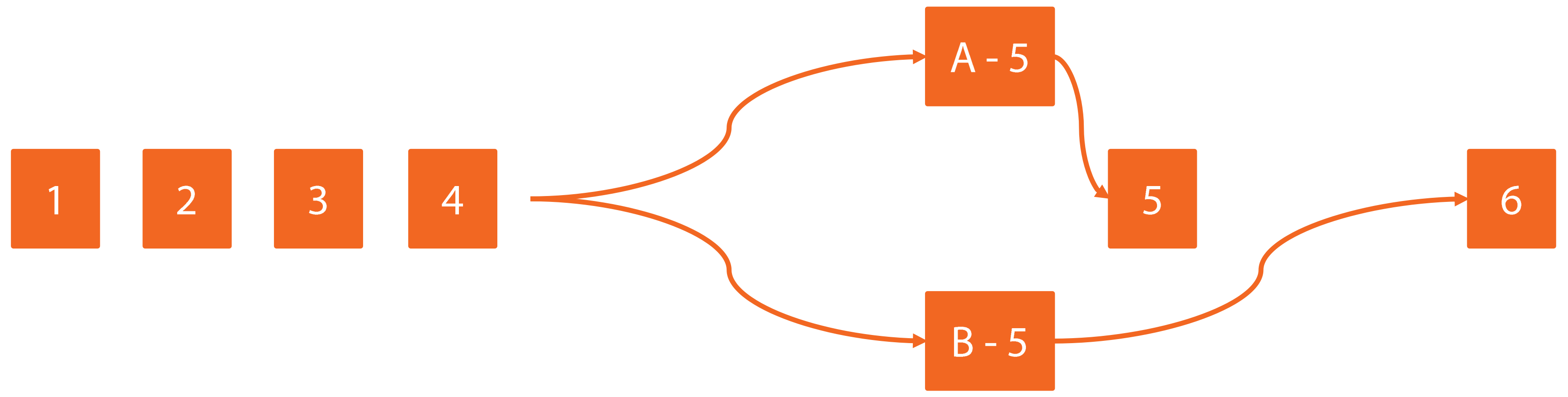
```
ALTER VIEW employee_positions AS
SELECT employees.id AS employee_id,
        name,
        title
FROM employees
        LEFT JOIN titles on employees.title_id = titles.id
```

LATER

DROP TRIGGERS

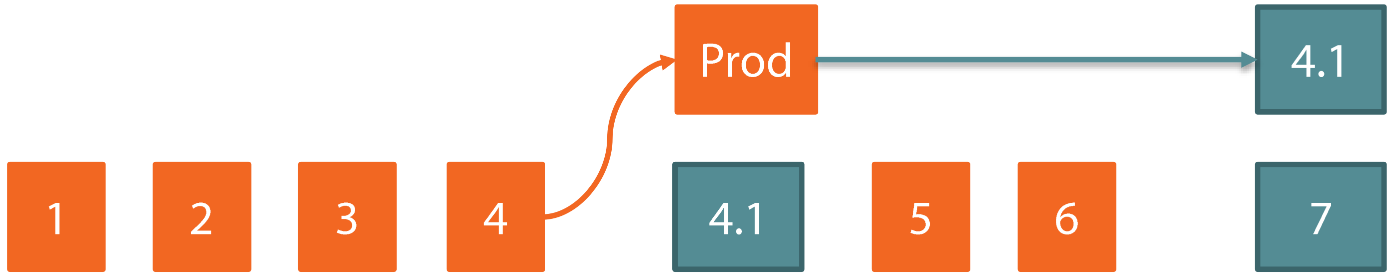
```
ALTER TABLE employees DROP column last_name
ALTER TABLE employees DROP column first_name
GO
```

Feature Branches



Release Branches

- 4.1 - flyway outOfOrder
- 7 - idempotent


Prod
4.1
1
2
3
4
4.1
5
6
7