Eclipse Che 7: New & Noteworthy

Florent Benoit
Eclipse Che
Activities that require configuration and setup:

- Onboarding on a new project
- Switching between different projects, technologies
- Fixing a maintenance branch
- Reviewing a pull request
- Upgrading tools, plugins, dependencies
- Trying out a new technology
Anyone, anytime can contribute to a project without installing software.
Consistency, reproducibility, and ease of access is a velocity factor
TRADITIONAL LAPTOP APPROACH

Shareable with all:
- Project sources
- Dependencies
- Developer Tools: language servers, debuggers, testing tools, security tools, etc...

Shareable with some:
- Commands
- Build and packaging tools
- Terminal

Hard to share with anyone:
- Operating system
- Web server / application server
- Database
- (All other runtime components)

A laptop makes it hard to share and secure everything a developer needs.
Defining a new kind of workspace

Eclipse Che Workspace

Source Code
IDE
Tools
Runtime Env

Main void function() { ... }
New Editor
New editor: Eclipse Theia

Containerizing the IDE: zero install and automate configuration

Extended Eclipse Theia, to provide a VSCode-like experience

Built-in:
- Languages Server Protocol
- Debug Adapter Protocol

Compatible with VSCode extensions
Customize with VSCode extensions

- Visual Studio Code extensions compatibility
- Extensions packaged with their dependencies
- Plug-in registry with predefined set of plug-ins
Enriched Developer Experience

- Keyboard Navigation: Command Palette
- Rich Editor
  - Find/Replace instances
  - Peek Definition
  - Outline
- Improved Debug
- Simple Git Plugin
- Layout customization
- Theming
- Port detection Plugin
Swappable Editor
Che 7: swappable editor
Devfile
The devfile provides easy-to-configure, highly reproducible definitions of portable developer environments.

It is a declarative abstraction of a replicable developer workspaces, which includes the runtime environments, the source code of the projects mapped to repositories and the tools, plugins and commands needed to code, build, test, run and debug a project.
Devfile Example

1. Project information
2. List of components of the workspace
3. Plugin component
4. Runtime image
5. Env variables to configure the container
6. End-points definition

---
apiVersion: 1.0.0
metadata:
  generateName: java-web-vertx-
  projects:
    - name: java-web-vertx
      source:
        type: git
        location: "https://github.com/che-samples/web-java-vertx"
  components:
    - type: chePlugin
      id: redhat/java/latest
    - type: dockerimage
      alias: maven
      image: quay.io/eclipse/che-java8-maven:nightly
      env:
        - name: JAVA_OPTS
          value: "-Duser.home=/home/user"
        - name: MAVEN_OPTS
          value: $(JAVA_OPTS)
      memoryLimit: 512Mi
  endpoints:
    - name: '8080/tcp'
      port: 8080
      mountSources: true
      volumes:
        - name: m2
          containerPath: /home/user/.m2
Manage Consistency
First class support of K8S, manage the complexity of developer environments take out the pain.

Easy to Integrate
Devfiles are easy to integrate with any tools. Developer environments can be created from anywhere at anytime.

Simple to Share
Devfile live with source code, are easy to modify, fork and share.

Extensible
Customize per task, with plug-in and developer preferences.
DEMO #1
Stack and Plug-ins
Registries
Stack and Plug-ins Registries

**Tools Registry**
- In-house deployment
- Define plug-ins
- Define IDEs

**Devfile Registry**
- Provide ready-to-use developer stacks
- Manage stacks
Registries in Eclipse Che

- Plug-in registry
- Devfile registry
- Workspace
Manage all plugins including VS Code extensions
Develop VS Code extensions
How to start?

- **Scaffolding: Yeoman generator**
  - `yo code`

- **Develop**
  - Intellisense from Eclipse Che IDE

- **Run**
  - Run inside Eclipse Che IDE

- **Debug**
  - Run inside Eclipse Che IDE
Hosted / development mode

Two instances of Theia
- Main instance to write plug-ins
- A new instance to run plug-ins
DEMO #2

Develop and debug VSCode extension

youtube
Publish VS Code extensions on Eclipse Che registries
Registries in Eclipse Che

plug-in registry

workspace

devfile registry
apiVersion: 1.0.0
metadata:
  name: my-che-workspace
components:
  -
    type: chePlugin
    id: redhat/java11/latest
apiVersion: 1.0.0
metadata:
  name: my-che-workspace
components:
- type: chePlugin
- reference: https://my-registry/plugins/florent/hello/0.1/meta.yaml
Create its own registry

Build a custom registry based on existing one

- Add / update / remove plug-ins
- Rebuild docker image
Create its own registry on github

Or use a simple github project
DEMO #3

VSCode extension with custom plugin registry

youtube
VS Code extensions and Eclipse Che
ECLIPSE CHE

BENEFITS

Eclipse Che Workspaces
plug-ins per workspace

Multi-user / auth
OpenID Connect / LDAP / Keycloak

Plug-in Registry
Global / per instance

Cloud Ready
Kubernetes / OpenShift
Custom namespace

@eclipse-che/plugin namespace

- Interact easily with Eclipse Che API

  [GitHub Link]

  ```typescript
  export namespace workspace {
    export function getCurrentWorkspace(): Promise<cheApi.workspace.Workspace>;
    export function getAll(): Promise<cheApi.workspace.Workspace[]>; // Added semicolon
    export function getAllByNamespace(namespace: string): Promise<cheApi.workspace.Workspace[]>; // Added semicolon
    export function getById(workspaceId: string): Promise<cheApi.workspace.Workspace>;
    export function create(config: cheApi.workspace.WorkspaceConfig, params: KeyValue): Promise<any>;
    export function update(workspaceId: string, workspace: cheApi.workspace.Workspace): Promise<any>;
    export function deleteWorkspace(workspaceId: string): Promise<any>;
    export function start(workspaceId: string, environmentName: string): Promise<any>;
    export function startTemporary(config: cheApi.workspace.WorkspaceConfig): Promise<any>;
    export function stop(workspaceId: string): Promise<any>;
    export function getSettings(): Promise<KeyValue>;
  }
  ```

- Simple code to use

  ```javascript
  import * as che from '@eclipse-che/plugin';
  ...
  await che.workspace.getCurrentWorkspace();
  ```
EXTERNAL PLUG-IN DEPENDENCIES

System Runtime dependencies / Language server protocol
Java LSP needs Java, PHP requires PHP, etc

Tools dependencies
$ apt-get install / yum install

Native libraries
ECLIPSE CHE

Standalone VS Code

- VS Code
  - Plugin1
  - PluginN

Java installed on your system

Eclipse Che with containers

- Main IDE Container
  - Plugin 3
  - JSON RPC

- Che IDE Endpoint
  - Java 11 container
    - Plugin 1
  - Che IDE Endpoint

- Che IDE Endpoint
  - Java 13 container
    - Plugin 2
How sidecar endpoint is working?

Che Remote Theia endpoint

```bash
FROM eclipse/che-theia-endpoint-runtime:next
RUN apk --no-cache add openjdk11 -repository http://dl-cdn.alpinelinux.org/alpine/edge/community
ENV JAVA_HOME /usr/lib/jvm/default-jvm/

FROM eclipse/che-theia-endpoint-runtime:next
RUN apk --no-cache add fortune
```
How sidecar endpoint is working?

Che Remote Theia endpoint: some limitations

- Needs to inherit from a dedicated Docker image
- Workaround with multi-staged builds
- Strong link between Theia image and sidecar images
  - (better to use same version of theia on main theia and sidecars)
Che Remote Theia endpoint: some limitations

- Needs to inherit from a dedicated Docker image
- Workaround with multi-staged builds
- Strong link between Theia image and sidecar images
  - (better to use same version of theia on main theia and sidecars)
Upcoming Eclipse Che 7.4: endpoint improvements

- Create a single all-in-one binary for this endpoint
  - Using nexe tool on Eclipse Che Theia remote package
- Copy this binary and run it
- Allow to use existing dockers image from docker registries
  - FROM docker.io/openjdk:14-alpine
- ⇒ no need to build images, use image name in plugin’s meta.yaml
Use a custom runtime for a VS Code extension: meta.yaml

```yaml
apiVersion: v2
publisher: redhat
name: vscode-xml
version: latest
type: VS Code extension
displayName: XML
title: XML Language Support by Red Hat
description: This VS Code extension provides support for creating and editing XML documents, based on the LSP4XML language server, running with Java.
icon: https://www.eclipse.org/che/images/logo-eclipseche.svg
repository: https://github.com/redhat-developer/vscode-xml
category: Language
firstPublicationDate: '2019-04-19'
spec:
  containers:
  - image: "docker.io/eclipse/che-remote-plugin-runner-java11:next"
    name: vscode-xml
    memoryLimit: "768Mi"
  extensions:
  - https://github.com/redhat-developer/vscode-xml/releases/download/0.7.0/vscode-xml-0.7.0-3205.vsix
```
DEMO #4

VSCode extension with custom runtime

youtube
Status
VS Code extensions

https://che-incubator.github.io/vscode-theia-comparator/status.html

<table>
<thead>
<tr>
<th>Name</th>
<th>Theta master</th>
<th>Theta v0.11.0</th>
<th>Theta v0.10.0</th>
<th>VSCode master</th>
<th>VSCode 1.38.0</th>
<th>VSCode 1.36.1</th>
<th>VSCode 1.31.1</th>
<th>VSCode 1.36.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakpoint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constructor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>condition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hitCondition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>logMessage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BreakpointsChangeEvent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>added</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>removed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>changed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CancellationToken</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isCancellationRequested</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>onCancellationRequested</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CancellationTokenSource</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constructor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>token</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cancel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dispose</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CharacterPair</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CharacterPair</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clipboard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>readText</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>writeText</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CodeAction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constructor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>edit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diagnostics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>command</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kind</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isPreferred</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CodeActionContext</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diagnostics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>only</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CodeActionKind</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constructor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>append</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>intersects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contains</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CodeActionProvider</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>provideCodeActions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CodeActionProviderMetadata</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>provideCodeActionKinds</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CodeLens</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constructor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>command</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Theia-vscodecov checker

https://github.com/theia-ide/theia-vscodecov

$ cd my-vscode-extension
$ npx theia-vscodecov
Conclusion
Get started with Eclipse Che

Sources
github.com/eclipse/che

Dev List
che-dev@eclipse.org

Docs
http://eclipse.org/che/docs

Mattermost
https://mattermost.eclipse.org

Try it out: https://che.openshift.io
Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.