Developing cross-platform application with rich GUI using QtWebEngine

—

QtCon, September 2, 2016

Victor Kropp


JET
BRAINS

jetbrains.com

3 major releases per year

2-3 minor updates each

up to **10** early preview builds

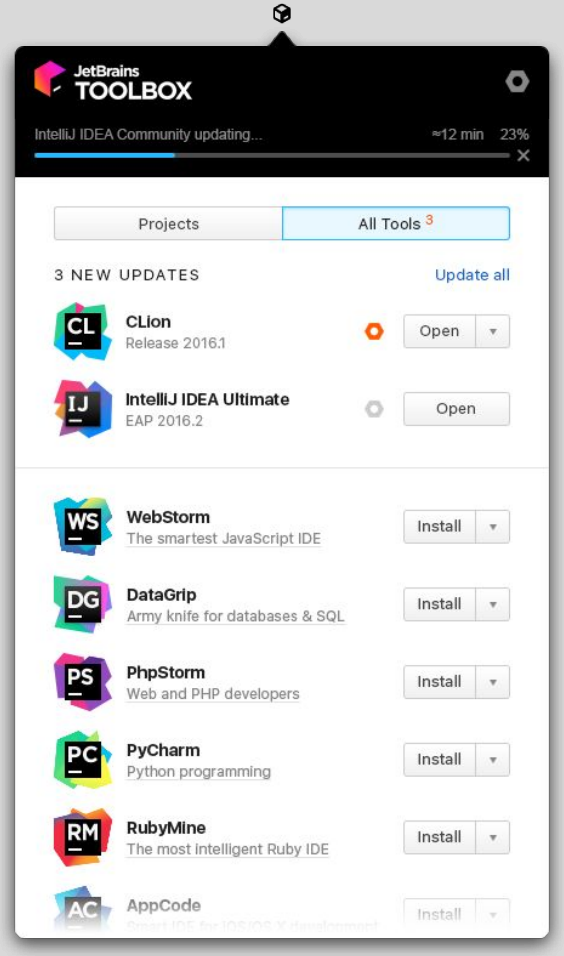
Toolbox App

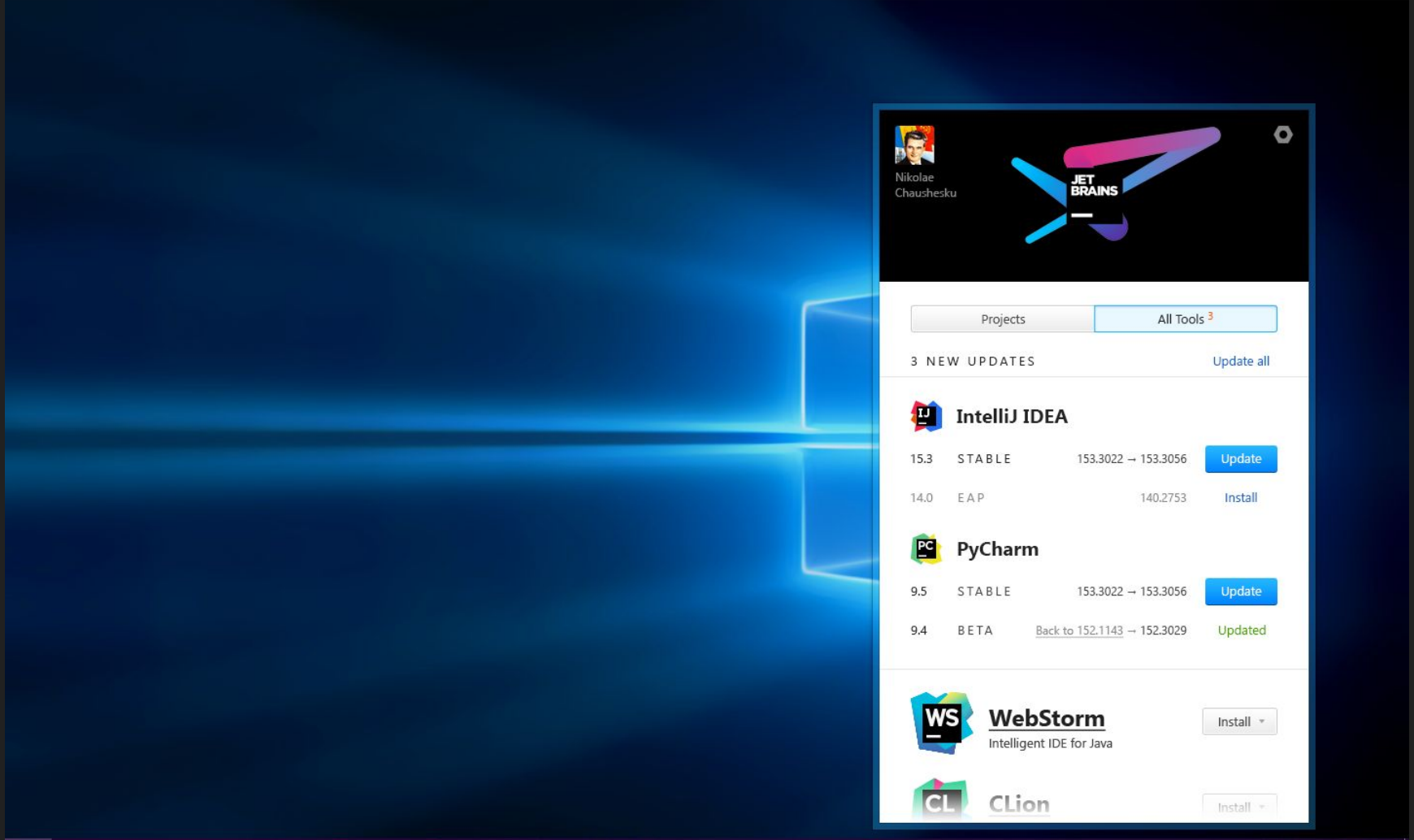
–

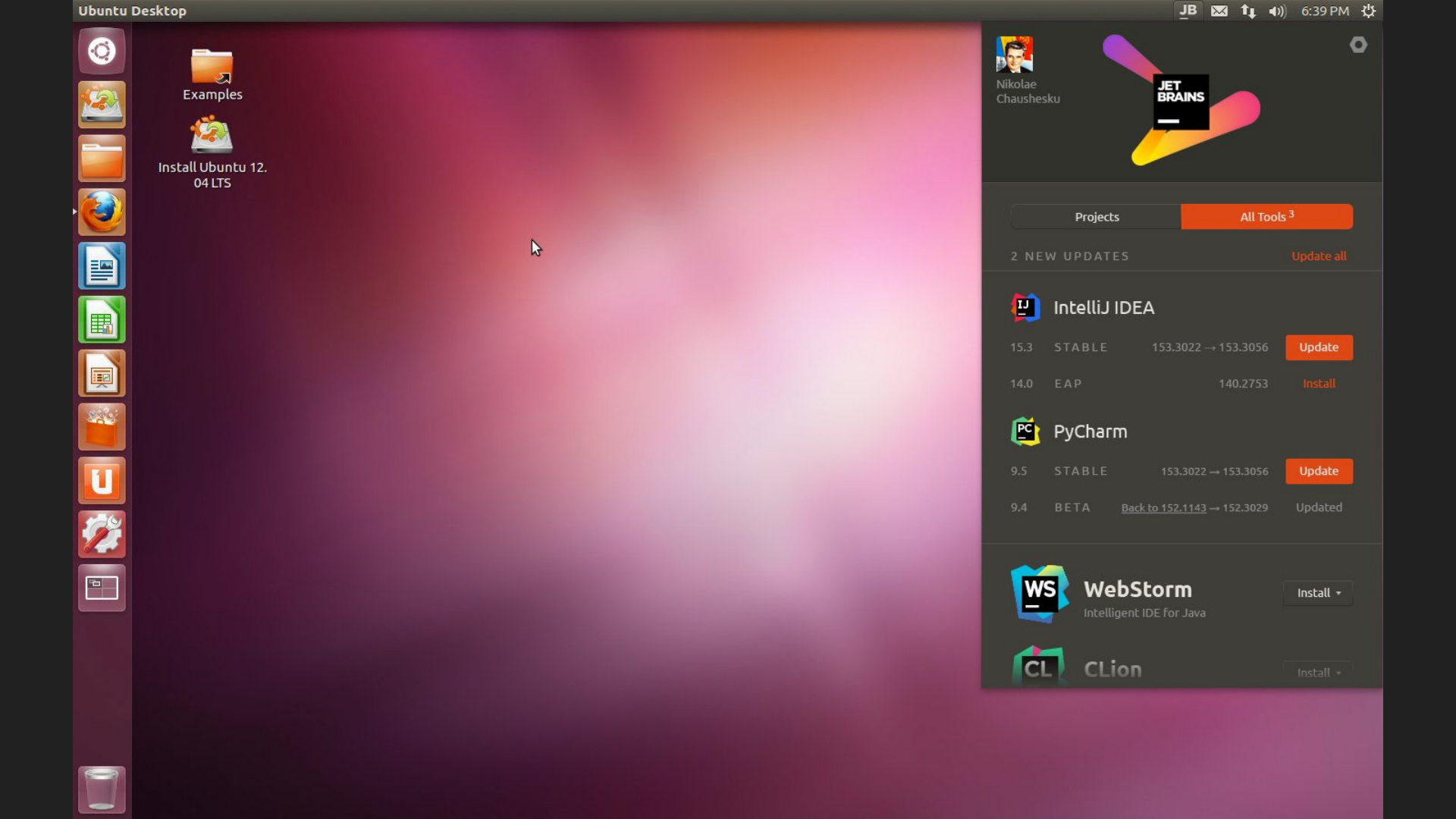
Recent projects

Installed & available tools

Easy install & update




Nikolae
Chaushesku
JET
BRAINS
Projects
All Tools 3
3 NEW UPDATES
Update all
IntelliJ IDEA
15.3 STABLE 153.3022 → 153.3056 Update
14.0 EAP 140.2753 Install
PyCharm
9.5 STABLE 153.3022 → 153.3056 Update
9.4 BETA Back to 152.1143 → 152.3029 Updated
WebStorm
Intelligent IDE for Java
Install
CLion
Install
JB
10:24 PM
7/2/2015


Ubuntu Desktop
JB
6:39 PM
Examples
Install Ubuntu 12.
04 LTS
Nikolae
Chaushesku
JET
BRAINS
Projects
All Tools 3
2 NEW UPDATES
Update all
IntelliJ IDEA
15.3 STABLE 153.3022 → 153.3056 Update
14.0 EAP 140.2753 Install
PyCharm
9.5 STABLE 153.3022 → 153.3056 Update
9.4 BETA Back to 152.1143 → 152.3029 Updated
WebStorm
Intelligent IDE for Java
Install
CLion
Install

Hackathon

–

Held annually

48 hours

Do whatever you want

Mixed teams

Team

—

UX Designer

UI Developer

Platform Developer

3 × Core Developers: Mac OS X, Windows & Linux

Disclaimer

–

This is our first experience

- in Qt
- in HTML/CSS/JS UI in Desktop application

HTML/JS Desktop Applications

–

Chrome Web Apps (discontinued)

Electron-based apps

http://electron.atom.io/apps/

Built on Electron

With Electron, creating a desktop application for your company or idea is easy. Initially developed for GitHub's Atom editor, Electron has since been used to create applications by companies like Microsoft, Facebook, Slack, and Docker.

Atom Slack Visual Studio Code Hive Avocode Nuclide

5EClient Theme Juice

Make a Pull Request to add your application.

Great tools from the community

Open source projects that help you build better apps.

- Electron Packager — Package your apps
- Electron Builder — Deploy your apps
- Spectron — Test your apps
- Devtron — Debug your apps
- Electron Prebuilt — Install Electron
- Menubar — Create menubar apps

View more community projects.

Why not Electron?

—

We didn’t want to write **that** much JS code

What did we choose?

—

Native cross-platform application in C++

Qt 5.6 + QtWebEngine

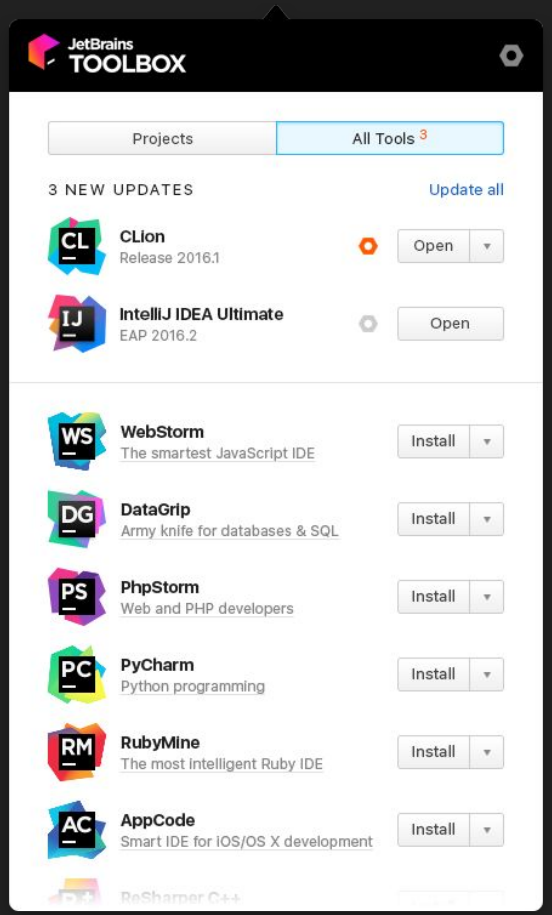
GUI built in JavaScript with React

How it works

–

Applications descriptions
are regularly updated via JSON feed

The same JSON is used in C++
and JS code



Feed JSON

—


```
{
   "id": "IDEA-U",   "name": "IntelliJ IDEA Ultimate",
   "description": "The most intelligent Java IDE",
   "icon_url": "…",
   "licensing": { "product": "Idea" },
   "build": "145.972.3",   "version": "2016.1.2",
   "major_version": { "name": "2016.1" },
   "package": {
       "os": "windows",
       "type": "nsis",       "command": "bin/idea.exe",
       "url": "https://download.jetbrains.com/idea/ideaIU-2016.1.2.exe",
       "size": 407042160,
       "checksums": [ { "alg": "sha-256", "value": "…" } ]
   }
}
```

MVVM

—

Qt/C++

- Model
- View

React/JavaScript

- View Model

Model

—

Available tools

Installed tools

Available updates

Detected projects

View

–

All combined in one JSON object

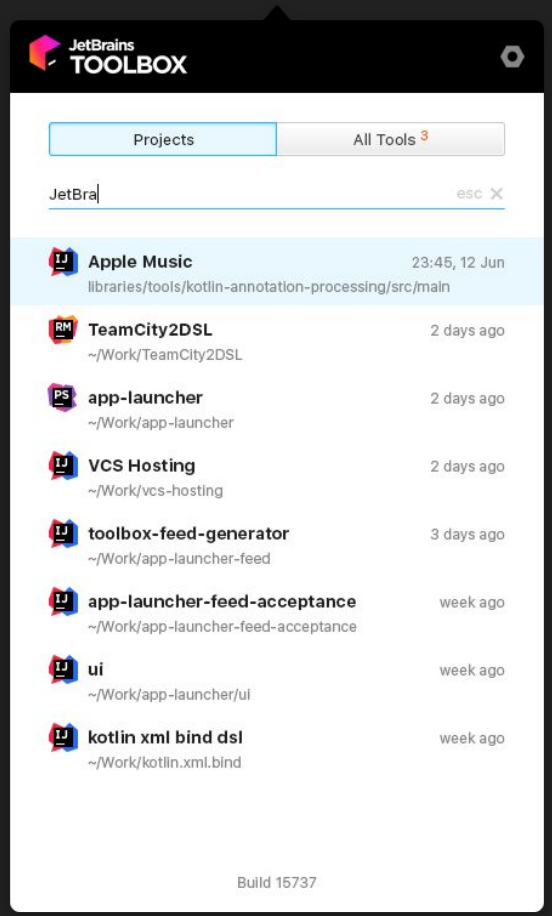
On every change a signal is emitted

ViewModel

—

Model JSON received as a signal parameter becomes the state of React’s component

Example


JetBrains
TOOLBOX
Projects
All Tools 3
JetBra
esc
Apple Music
23:45, 12 Jun
libraries/tools/kotlin-annotation-processing/src/main
TeamCity2DSL
2 days ago
~/Work/TeamCity2DSL
app-launcher
2 days ago
~/Work/app-launcher
VCS Hosting
2 days ago
~/Work/vcs-hosting
toolbox-feed-generator
3 days ago
~/Work/app-launcher-feed
app-launcher-feed-acceptance
week ago
~/Work/app-launcher-feed-acceptance
ui
week ago
~/Work/app-launcher/ui
kotlin xml bind dsl
week ago
~/Work/kotlin.xml.bind
Build 15737

Example: Model

```
class RecentProject {
  private:
    QString myName;
    QString myFullPath;
    QString myDisplayPath;
    QDateTime myLastModified;
    RemoteFeedItem myFeedItem;

  public:
    explicit RecentProject(QString name, ...);

    QJsonObject toJson() const;
};
```

Example: View

```
signals:
  void recentProjectsDetected(QVariantList projects, QPrivateSignal);

public slots:
  void detectRecentProjects();
  QVariantList getRecentProjects();
```

Example: ViewModel

—

```
class ProjectList extends React.Component {
 render() {
   return (
     <div> <div className="project-list__search">
         <Icon size={Icon.Size.Size14} glyph={require('search.svg')} />
         <TextInput ref="input" placeholder="Search" />
       </div> <div>
         {recentProjects.map((project) => (
           <Project project={project} />
         ))}
     </div> </div>
   );
 }
}
```

How we are building it

CMake

–

Native for **CLion**

Supports Qt

Objective-C/C++ support

Objective-C

—

.plist read/write

Menubar icon theming

C++11

—

```
QtConcurrent::run([&]() {
  ...
});
```

```
connect(mySettingsAction, &QAction::triggered,
        uiApi, &UIApiConnector::showSettingsPage);
```

Testing

–

Started with QTest

Switched to Google Test

Inversion of Control

—

≃ 35 components

Dependency Injection manually in `main()`

Tips & Tricks

–

Do **NOT** use `#ifdef` for OS-specific things

Design for forward and backward compatibility

Dogfooding

Frontend

–

React + Redux

EcmaScript 6

Webpack

Frontend: React

–

React

is a declarative JavaScript library for building user interfaces.

https://facebook.github.io/react/

Frontend: Redux

–

Redux

is a predictable state container for JavaScript apps.

http://redux.js.org/

ECMAScript 2015

—

aka ECMAScript 6

Compiled with Babel into ECMAScript 5

Supported natively in Chrome 52, so will be available in **Qt 5.8**

Webpack

—

JS + dependencies are packed into single `index.js`

and `index.html`:

```
myWebView->load(QUrl("qrc:/index.html"));
```

index.html

```
<!DOCTYPE html>
<html><head>
 <title>JetBrains Toolbox</title>
 <script type="text/javascript"
src="qrc:///qtwebchannel/qwebchannel.js"></script>
</head><body>
 <div id="app-container"></div>
 <script
src="{%=o.htmlWebpackPlugin.files.chunks.index.entry%}"></script>
</body></html>
```

Development mode

—

Real `index.html` is added during the build

In development we instead use

```
myWebView->load(QUrl("http://0.0.0.0:8080/"));
```

JS API

—

Single QObject for JS API:

```
myChannel = new QWebChannel(this);
myChannel->registerObject("api", api);
myWebView->page()->setWebChannel(myChannel);
```

GUI API Stubs (in TypeScript)

```
class onWindowShown { connect(callback: () => void): void; }
class refreshApplicationsList {
    connect(callback: (model: Object) => void): void;
}
class Api {
    onWindowShown: onWindowShown;
    refreshApplicationsList: refreshApplicationsList;
    openSystemProxySettings(): void;
    getRecentProjects(callback: (result: Array) => void): void;
}
class WebChannelObjects { api: Api; }
class WebChannel { objects: WebChannelObjects; }
```

Security

–

HTTPS only

Signed feed

Checksum for all downloaded packages

Installers

–

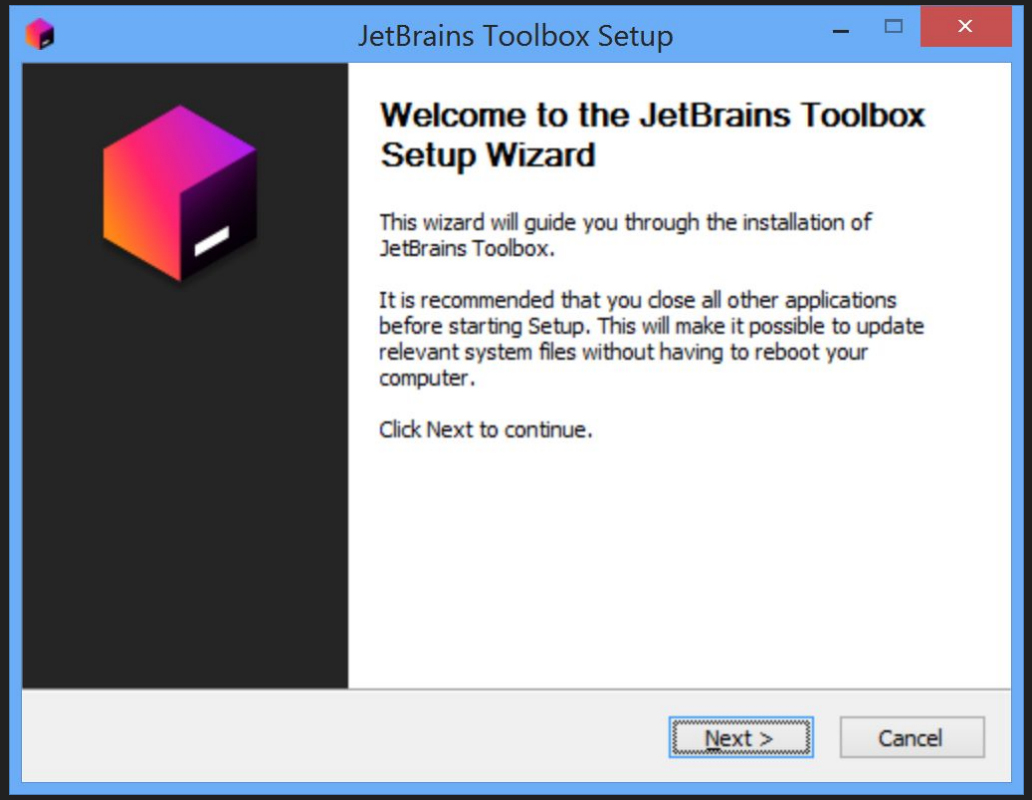
CPack for Windows/Mac OS X

Simple Shell script on Linux

Windows: NSIS

```
if(WIN32)
 qt5_use_modules(LIB WinExtras)
 set(CPACK_GENERATOR NSIS)
 set(CPACK_NSIS_EXECUTABLE_FILE_NAME "${EXECUTABLE_NAME}.exe")
 set(CPACK_NSIS_MUI_ICON "${CMAKE_CURRENT_SOURCE_DIR}/toolbox.ico")
 set(CPACK_PACKAGE_INSTALL_REGISTRY_KEY "JetBrainsToolbox")
 set(CPACK_NSIS_ENABLE_UNINSTALL_BEFORE_INSTALL "ON")
 set(CPACK_NSIS_DISPLAY_NAME "JetBrains Toolbox")
 set(CPACK_NSIS_URL_INFO_ABOUT "https://www.jetbrains.com")
 set(CPACK_PACKAGE_INSTALL_DIRECTORY
"${ORGANIZATION_NAME}\\\\${APPLICATION_NAME}\\\\bin")
 include(CPack)
endif(WIN32)
```

Windows: NSIS

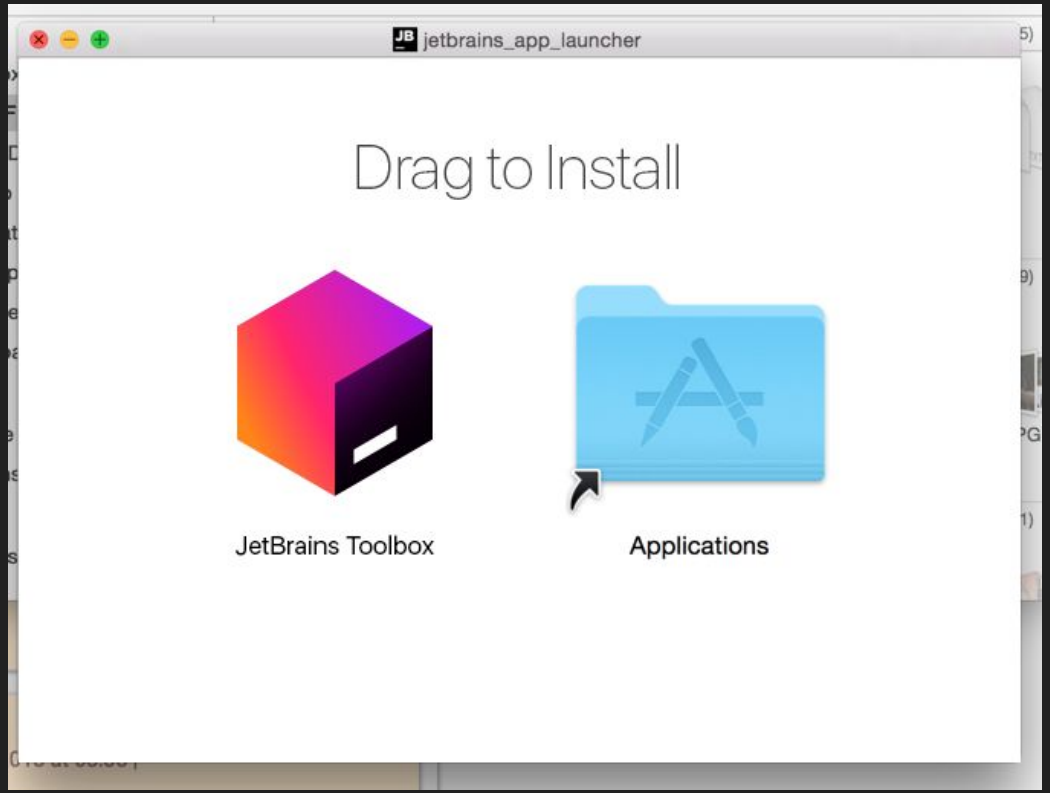


Mac OS X: OSA Script

—

```
tell application "Finder"
 tell disk "'${title}'"
  open
  set current view of container window to icon view
  set toolbar visible of container window to false
  set icon size of theViewOptions to 128
  set background picture of theViewOptions to file ".skin:skin.tiff"
  set position of item "'${applicationName}'" of container window to {100, 120}
  set position of item "Applications" of container window to {340, 120}
  set position of item ".DS_Store" of container window to {9375, 106}
  set bounds of container window to {400, 100, 855, 395}
  set position of container window to {155, 155}
  close
 end tell
end tell
```

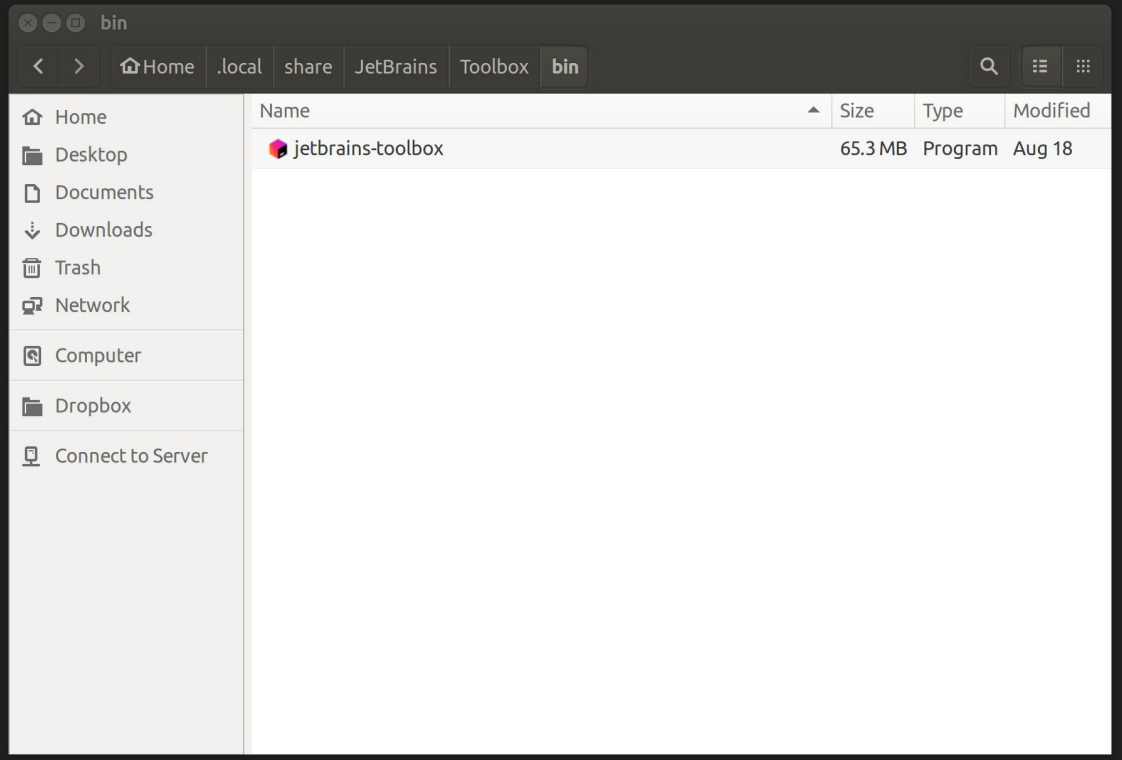
Mac OS X: DMG


jetbrains_app_launcher
Drag to Install
JetBrains Toolbox
Applications

Linux: AppImage

```
mkdir -p $APPDIR/usr/share/$APP
cp $APP $APPDIR/usr/share/$APP
cp ../jetbrains-logos/toolbox.svg $APPDIR/usr/share/$APP
cp ../deps/openssl-linux/lib/lib*.so* $APPDIR/usr/share/$APP
# QT binaries
QTLIBS="Core Widgets Gui Network Xml WebEngine WebEngineWidgets WebSockets
WebChannel WebEngineCore Quick Qml XcbQpa DBus"
for QTLIB in $QTLIBS; do
 cp -P $QTPATH/lib/libQt5$QTLIB.so $APPDIR/usr/share/$APP
done
# Make AppImage
$APPIMAGEASSISTANT $APPDIR $APP
chmod a+x $APP
```

Linux: AppImage



Thank you for your attention!

@JBToolbox
jb.gg/toolbox-app

Questions?

@JBToolbox
jb.gg/toolbox-app

Thank you!

jetbrains.com