

Projector: Your Remote IDE

Pasha Finkelshteyn, JetBrains

Did you ever complain?

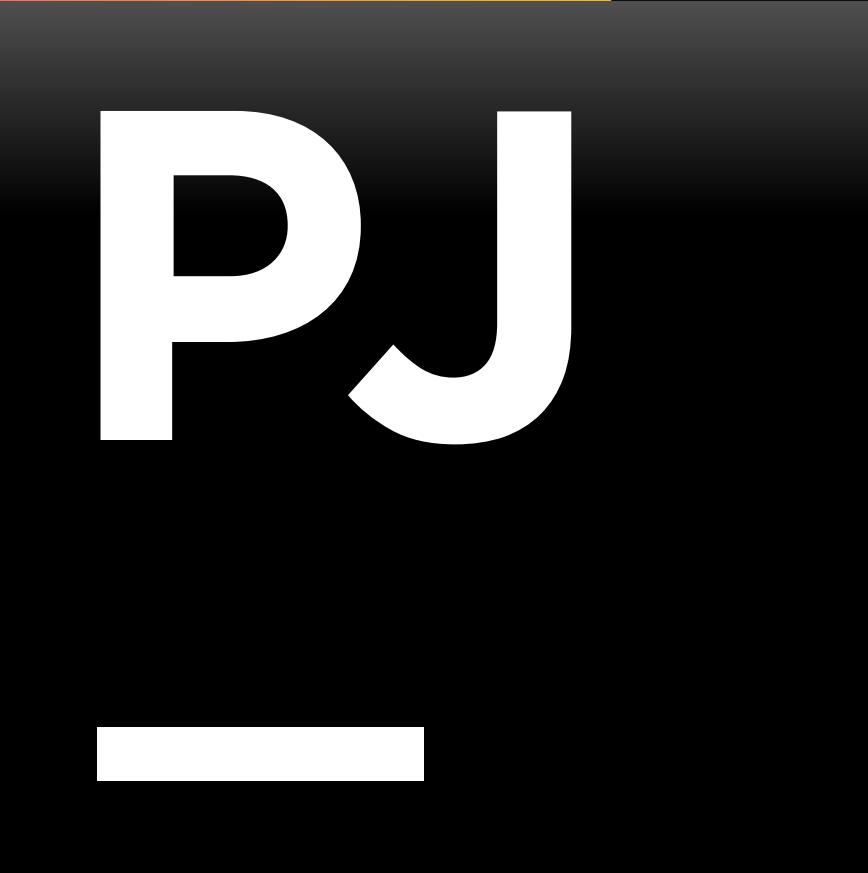
- You need to set up a VPN to access data?
- You need to use RDP to access your instance of IDEA?
- Or even VNC?
- *Long* indexing?
- Huge project compiles forever?



Did you ever want?

- Edit code from tablet/smartphone?
- Help your colleague to fix some issues?
- Show demo which accesses data from your cloud?
- Work from a weak laptop?





PJ

Issues with existing solutions. VPN

1. OpenVPN does not scale
 1. No central identity management
 2. No failover OOTB
2. Proprietary solutions are... proprietary
3. And expensive sometimes
4. IKE is not completely cross-platform

Issues with existing solutions. VNC

1. **Extremely** slow
2. Complex infrastructure (VPN?)

Issues with existing solutions. RDP

RDP is awesome

1. Single-user
2. Requires special client
3. Still needs complex infrastructure to provide access

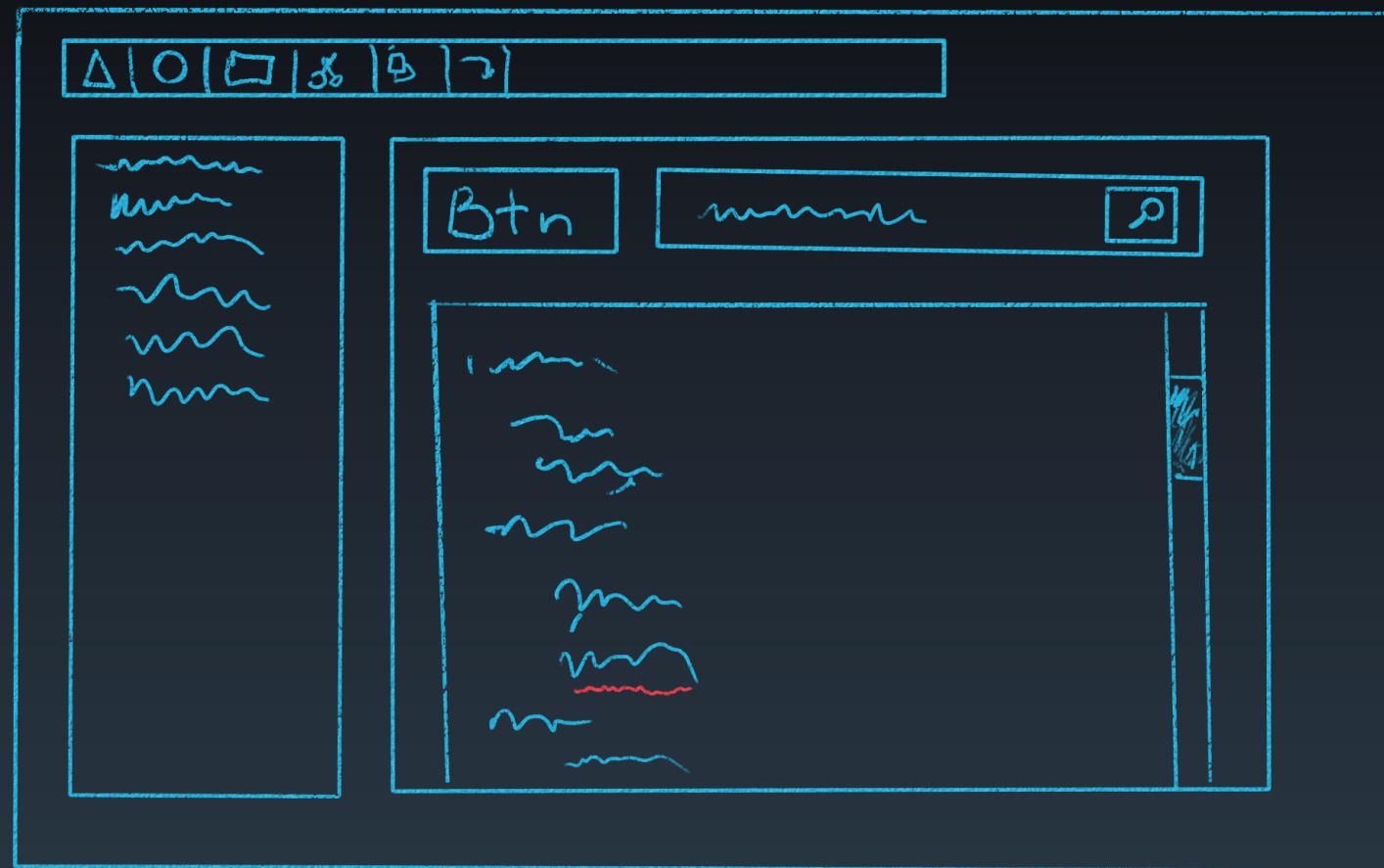
Stop this

MARKETING

Let's dig!



Typical Swing interface

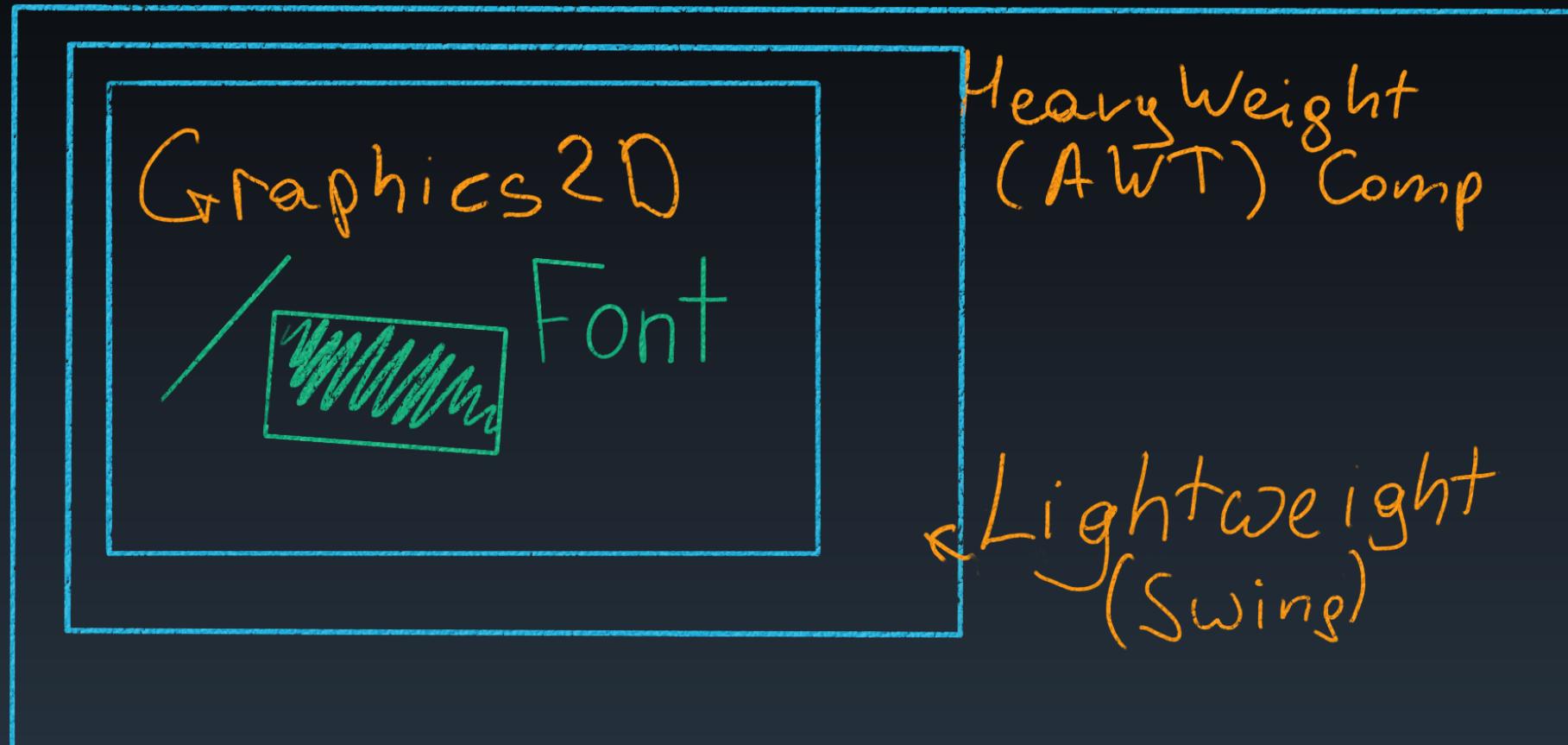


And in reality?

- Lines
- Simple shapes
- Strings
- Fonts

Swing architecture

Toolkit



Definitions

Toolkit – abstraction over the whole graphics

Heavyweight components – system-controlled components

Lightweight components – user-controlled components

Graphics2D – drawing controller

Types of drawing surfaces

- Visible
- Invisible

Part of the editor, which you don't see, is separate and is drawn on an invisible surface.

Each window has its surface(s)

Heavyweight components

There is a corresponding `Peer` class for each heavyweight component.

```
interface ComponentPeer {  
    void paint(Graphics g);  
    void setBounds(int x, int y, int width, int height, int op);  
    void handleEvent(AWTEvent e);  
    void setBackground(Color c);  
    void setFont(Font f);  
    // etc.
```

Responds for delegation to Graphics and setting several params

Heavyweight components

We have implementations of peers only for several heavyweight components.

Graphics2D

Anything may contain Graphics2D

Responds for drawing:

```
abstract class Graphics2D {  
    public abstract void drawImage(BufferedImage img,  
                                    BufferedImageOp op,  
                                    int x, int y);  
    public abstract void drawString(String str, int x, int y);  
    public abstract void fill(Shape s);  
    // etc.
```

How the hell will it work in a browser?



How the hell will it work in a browser?

The initial version just sent the whole screen to the client.

It had a whopping performance of ≈5 fps 😱



How the hell will it work in a browser?

Welcome [DrawEventQueue](#)

Queue that will transform any action into an event, based on

```
ConcurrentLinkedQueue<List<ServerWindowEvent>>()
```

DrawEventQueue

Each component has its queue (for now)

```
private inline fun paintArea(  
    crossinline command: DrawEventQueue.CommandBuilder.() -> Unit  
) {  
    drawEventQueue  
        .buildCommand()  
        .setClip(identitySpaceClip = identitySpaceClip)  
        .setTransform(transform.toList())  
        .setComposite(backingComposite)  
        .command()  
}
```

Images

There are at least five types of images in AWT



VolatileImage :

- is used to display off-screen surfaces
- is drawn by primitives
- is being sent as commands.

Images

BufferedImage :

- is being sent as bitmap
- we have sophisticated reflection-based heuristics to determine if it is changed
- may be optimized in future

Images

MultiresolutionImage :

- used for drawing icons
- supported only if it has the single resolution

Images

Other types of images are not supported right now.

So we have commands. Now what?

```
while(!stopped){  
    sendAllCommands() // sends both images and command  
    Thread.sleep(10)  
}
```

Every 10ms (roughly), aggregate all commands and send them to the client over WebSocket.

Format

Default format: highly optimized JSON

Supported OOTB formats: Protobuf, JSON

May support anything if we'll implement it on both sides.

Format

```
[["e", {"a": ["a", {"a": 2}], "b": [{"h": {"a": ["a", {"a": 252.0, "b": 78.0, "c": 26.0, "d": 24.0}]]}],  
["p", {"a": [1.0, 0.0, 0.0, 1.0, 21.0, 76.0]}], ["i", {"a":  
["a", {"a": 1.0, "b": "a", "c": "c", "d": 10.0, "e": 0.0, "f": null}]},  
["s", {"a": ["a", {"a": -12828863}]}], ["r", {"a": ["a", {"a": "a", "b": 1.0}]}],  
["d", {"a": "b", "b": 0.0, "c": 0.0, "d": 478.0, "e": 28.0}], ["p", {"a":  
[1.0, 0.0, 0.0, 1.0, 252.0, 78.0]}], ["l", {"a": ["a", {"a": 1024, "b": -172147019}], "b":  
["b", {"a": 5, "b": 4, "c": 16, "d": 16, "e": null}]}]]}]
```

Everything is determined in runtime – types, fields, etc.
For example, "e" at the very beginning is type!

Backend

- Ktor?
- Tomcat?
- Undertow?
- Netty?
- Jetty?

Move!

GPL is not always a good thing

Popular products are not compatible with it, and we HAVE TO publish source code under GPL license.

So...

[TooTallNate/Java-WebSocket](#)

And a bunch of crutches to add HTTP support there.

Clients

A client should implement ≈ 30 API methods of drawing + request resources

The excellent part of Kotlin: we can share some logic between client and server (and we do).

`kotlinx.serialization` *may* do things more straightforward (but deserialization is sooo slow).

A non-read-only client should be able to send mouse and keyboard events, and so on.

In browser

```
879     public void should_fail_when_elements_are_not_transitively_equal() {
880         final VerificationInstancesCreator<TestComparable> lesser =
881             VerificationInstancesCreators.from(
882                 new LesserTestComparable());
883         final VerificationInstancesCreator<TestComparable> equal =
884             VerificationInstancesCreators.from(
885                 new EqualTestComparable(initializer: 1),
886                 new EqualTestComparable(initializer: 2),
887                 new EqualTestComparable(initializer: 7));
888         final VerificationInstancesCreator<TestComparable> greater =
889             VerificationInstancesCreators.from(
890                 new GreaterTestComparable());
891         try {
892             lesser.equals(equal);
893             greater.equals(equal);
894             greater.equals(greater);
895             lesser.equals(greater);
896             equal.equals(greater);
897             fail("Expected exception");
898         } catch (AssertionError e) {
899             assertEquals("equals(1) equals to equals(7), equals(2) equals to equals(7), but equals(1) is not equal to equals(2)", e.getMessage());
900         }
901     }
902     @Test
903     public void should_pass_when_elements_are_transitively_equal() {
904         final VerificationInstancesCreator<TestComparable> lesser =
905             VerificationInstancesCreators.from(
906                 new LesserTestComparable());
907         final VerificationInstancesCreator<TestComparable> equal =
908             VerificationInstancesCreators.from(
909                 new EqualTestComparable(initializer: 1),
910                 new EqualTestComparable(initializer: 2),
911                 new EqualTestComparable(initializer: 7));
912         final VerificationInstancesCreator<TestComparable> greater =
913             VerificationInstancesCreators.from(
914                 new GreaterTestComparable());
915         try {
916             lesser.equals(equal);
917             equal.equals(greater);
918             greater.equals(equal);
919             lesser.equals(greater);
920             equal.equals(greater);
921         } catch (AssertionError e) {
922             fail("Expected no exception");
923         }
924     }
925 }
```

```
92     .suppressConsistentWithEquals(true)
93     verify()
94     [REDACTED]
95   
```

```
</pre>

```

jector-launcher

```
* Please be aware that some of the checks done by this class expect that the
* instances have a {@link Object#toString()} implementation. This is very
* important as it is used for creating assertion messages.
*
* @param <A> type of the class under test
* @see Comparable
* @see VerificationInstancesCreator
* @see VerificationInstancesCreators
*/
public final class ComparableVerifier<A extends Comparable<A>> {
    private final VerificationInstancesCreator<A> lesserCreator;
    private final VerificationInstancesCreator<A> greaterCreator;
    private final VerificationInstancesCreator<A> equalCreator;

    private boolean suppressConsistentWithEquals = false;
    private boolean suppressEqualsToNullReturnsFalse = false;
    private boolean suppressExceptionOnCompareToNull = false;

    private ComparableVerifier(final VerificationInstancesCreator<A> lesserCreator,
                               final VerificationInstancesCreator<A> equalCreator,
                               final VerificationInstancesCreator<A> greaterCreator) {
        this.lesserCreator = lesserCreator;
        this.greaterCreator = greaterCreator;
        this.equalCreator = equalCreator;
    }

    /**
     * Creates an instance of the {@link ComparableVerifier}.
     *
     * @param lesserCreator "lesser" instances factory
     * @param equalCreator "equal" instances factory
     * @param greaterCreator "greater" instances factory
     * @param <A> type of the class under test
     * @return instance of {@link ComparableVerifier}
     */
}
```

Desktop client

- Electron-based
- (Almost) all browser hotkeys are disabled
- May be expanded to full screen

Installation: GNU/Linux

```
pipx install projector-installer
```

```
> projector run
    1. DataGrip
    2. IntelliJ
    3. IntelliJ_IDEA_Toolbox_ch-0
Choose a configuration number or 0 to exit: [0-3]: 1
Configuration name: DataGrip
To access IDE, open in browser
    http://localhost:9876/?token=qwertyuiop%21%40%23%24%25%5E%26%2A%28%29
    http://127.0.0.1:9876/?token=qwertyuiop%21%40%23%24%25%5E%26%2A%28%29
    http://192.168.1.111:9876/?token=qwertyuiop%21%40%23%24%25%5E%26%2A%28%29

To see Projector logs in realtime run
    tail -f "/home/finkel/.projector/configs/DataGrip/projector.log"

Exit IDE or press Ctrl+C to stop Projector.
```

Configuration

```
> projector config edit
  1. DataGrip
  2. IntelliJ
  3. IntelliJ_IDEA_Toolbox_ch-0
Choose a configuration number or 0 to exit: [0-3]: 1
Edit configuration DataGrip
Enter the path to IDE (default: /home/finkel/.projector/apps/DataGrip-2020.3.2, <tab> for complete):
Enter a Projector listening port (press ENTER for default) [9876]:
Would you like to specify listening address (or host) for Projector? [y/N]
Would you like to specify custom DNS names for Projector access? [y/N]
Use secure connection (this option requires installing a projector's certificate to browser)? [y/N]
Would you like to set password for connection? [y/N]
.
```

Installation: Docker

[JetBrains/projector-docker](#)

```
docker pull \  
    registry.jetbrains.team/p/prj/containers/projector-idea-u  
docker run --rm -p 8887:8887 -it \  
    registry.jetbrains.team/p/prj/containers/projector-idea-u
```

Useful scripts

- Build a container with IDE of your choice

```
build-container.sh [containerName [ideDownloadUrl]]
```

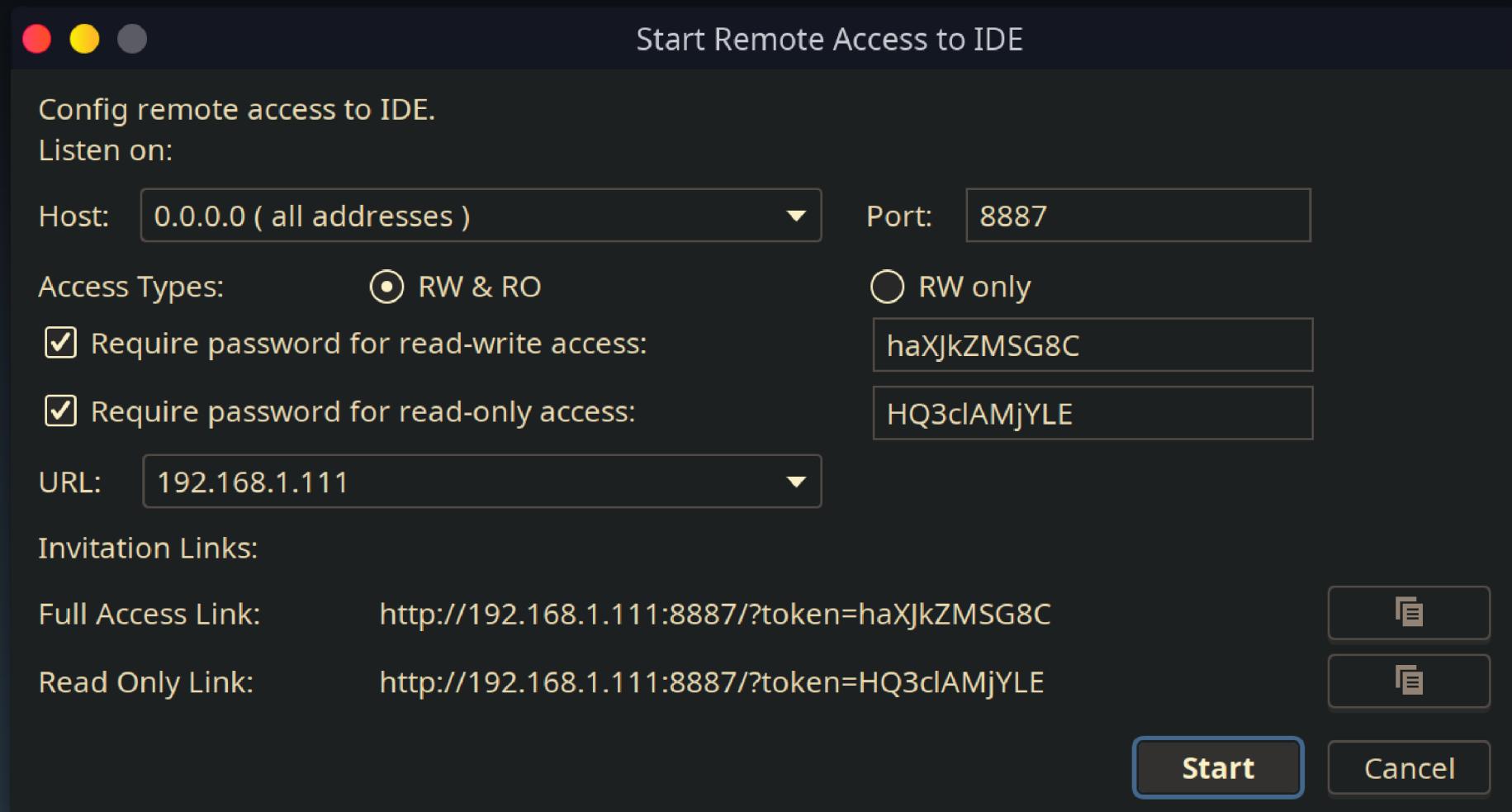
- Run Stateless container

```
run-container.sh [containerName]
```

- Run container with mounted directory to preserve state

```
run-container-mounted.sh [containerName]
```

Installation: IDE



How does it work?

```
$DRAW_HANDLER_CLASS_LOADING
clazz
.getMethod(
    "handleGraphics2D",
    new Class[] { String.class, Object[].class, java.awt.Graphics.class })
.invoke(
    null,
    new Object[] { "${it.longName}", $JAVASSIST_ARGS, $JAVASSIST_THIS});
```

Adds one more call to every drawing method

How does it compare to CWM?

- It's not a tool for collaborative development (for now)
- Allows displaying any IDE component
- Very comfortable to help/comment on somebody's issues

Place for improvement

- Visual glitches
- Support for (more) AWT components
- Improve HiDPI support
- Speculative typing (?)
- iOS (Android?) native clients (Flutter?)
- Queue optimization
- **You** name it

What did we learn?

- UI operations can be buffered
- There are not so many REAL UI operations
- Cross-platform drawing is achievable
- Parts of code may be reused between server and client

projector.jetbrains.com

Thank you

 asm0di0

 asm0dey

 newpodcast2.live