UNIVERSITÄT
BERN

P2 - Lab 02

Assignment 01

Two approaches

Regular expressions

Custom solution

Regular expressions

Did you escape all special characters?

Regular expressions

```
assertFalse(
  newFileFilter("f.\\.]+").
    accept(new File("fg.]+")));
```

Custom solution

Do you have a recursive solution?

Custom solution

```
assertTrue(
  newFileFilter("*anas").
    accept(new File("ananas")));
```

Custom solution

```
assertTrue(
  newFileFilter("*anas").
    accept(new File("ananas")));
```

```
assertTrue(
  newFileFilter("*nas*anas").
    accept(new File("ananasananas")));
```

(Not so perfect)

stuff

```
public class FilePattern implements FileFilter{

   public String string;

   public FilePattern(String string) {
      this.string = string;
   }


   ...
}
```

```
public class FilePattern implements FileFilter{

   public String string;

    public FilePattern(String string) {
        this.string = string;
    }


   ...
}
```

```
public class FilePattern implements FileFilter{

   private String string;

    public FilePattern(String string) {
        this.string = string;
    }


    ...
}
```

Make attributes private

```
public class FilePattern implements FileFilter{

    public String string;

    public FilePattern(String string ) {
        this.string = string;
    }


    . . .
}
```

```
public class FilePattern implements FileFilter{

    public String pattern;

    public FilePattern(String _pattern ) {
        this.pattern = _pattern;
    }


    . . .
}
```

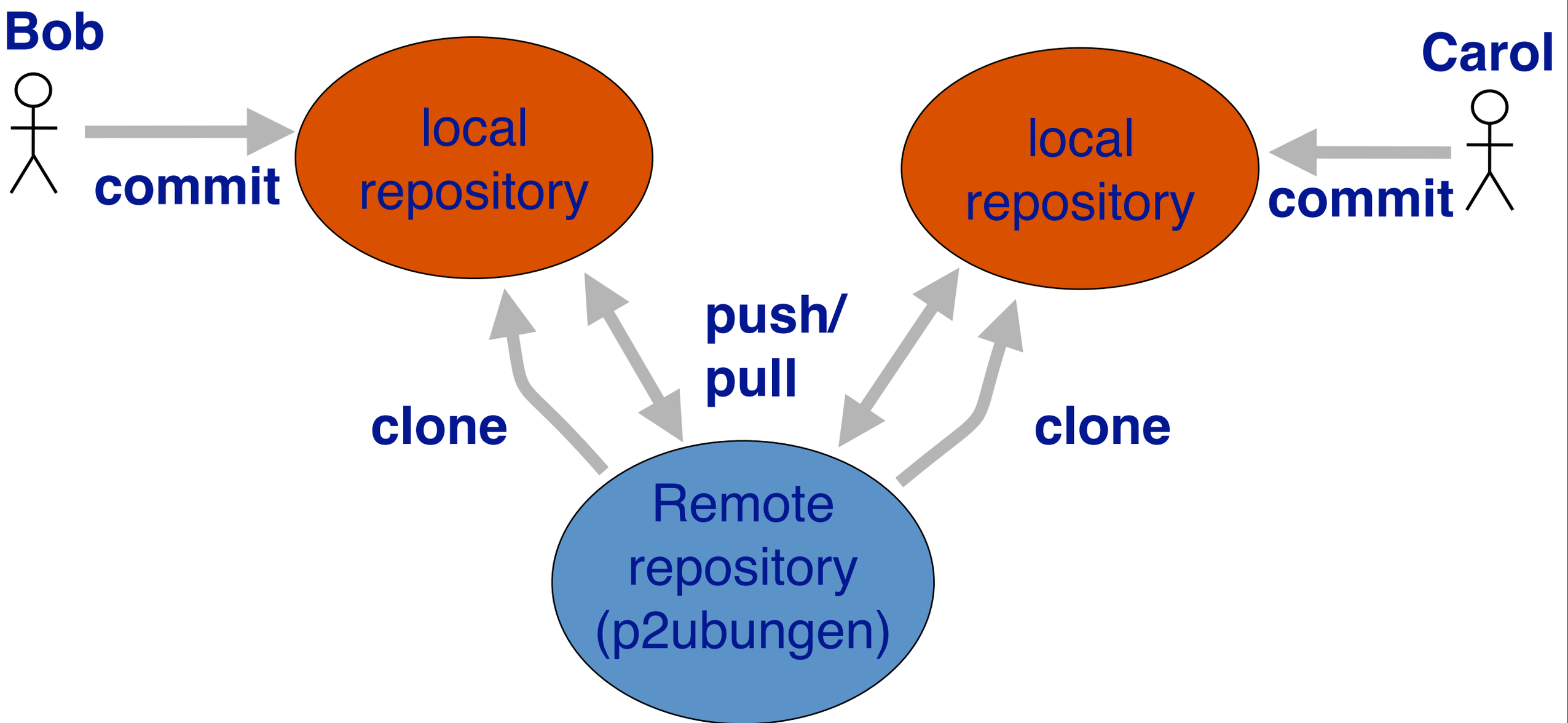
Use meaningful names

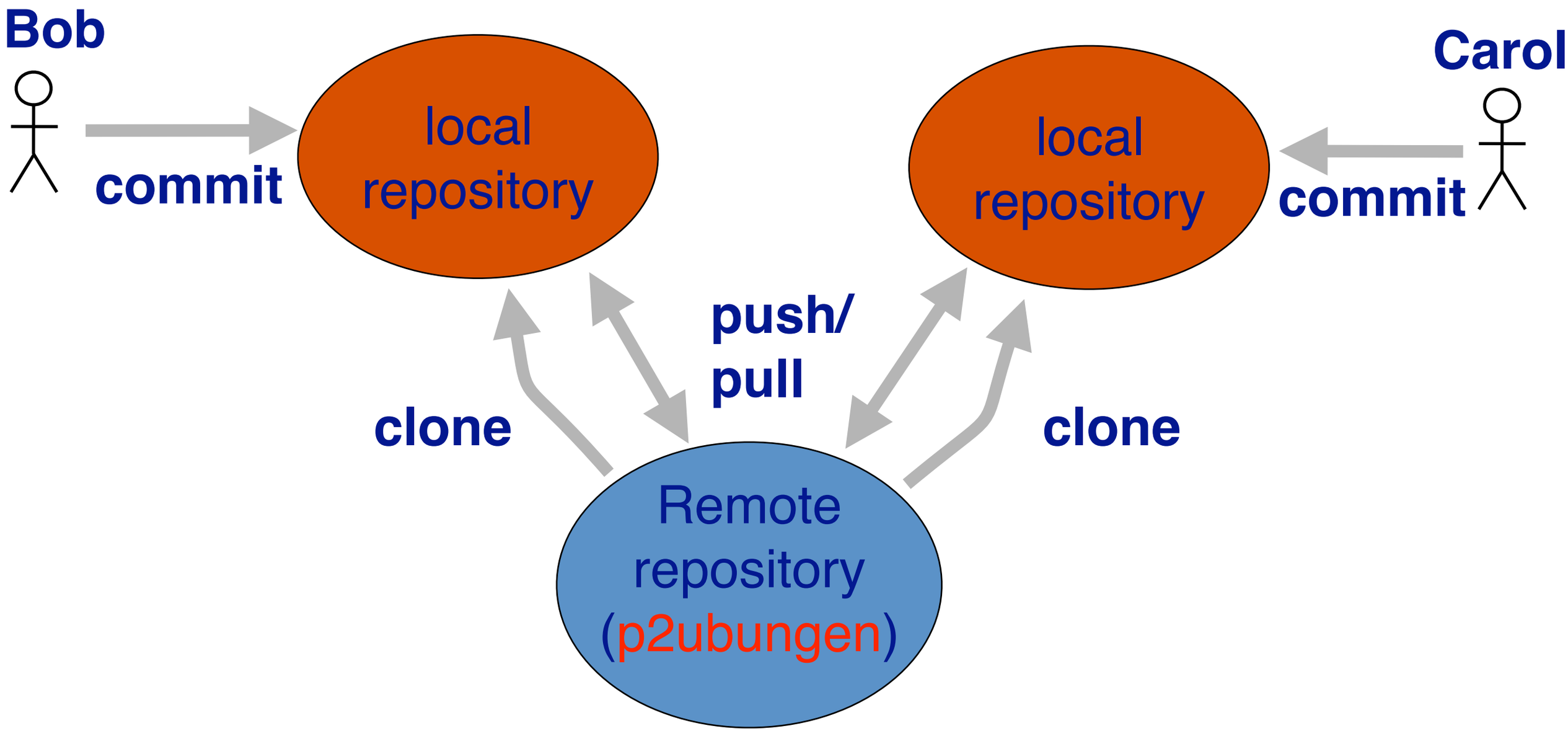
```
public boolean accept(File pathname) {

   ...

   // Check if name and string are the same
   if (string.matches(name))
      return true;
   else if(!string.matches(name))
      return false;

   // TODO END
   throw new Error();
}
```

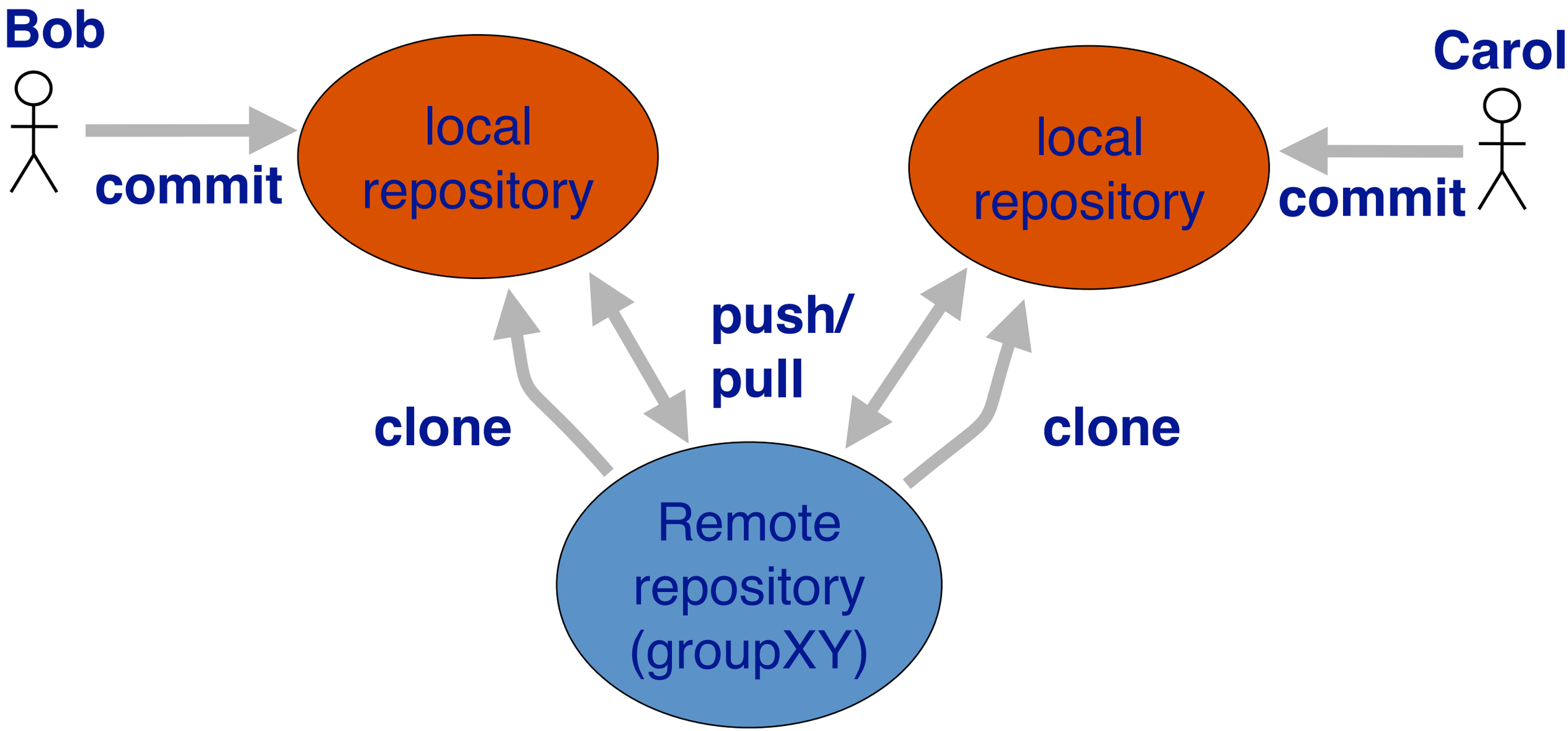
```
public boolean accept(File pathname) {

   ...

      return string.matches(name)
}
```

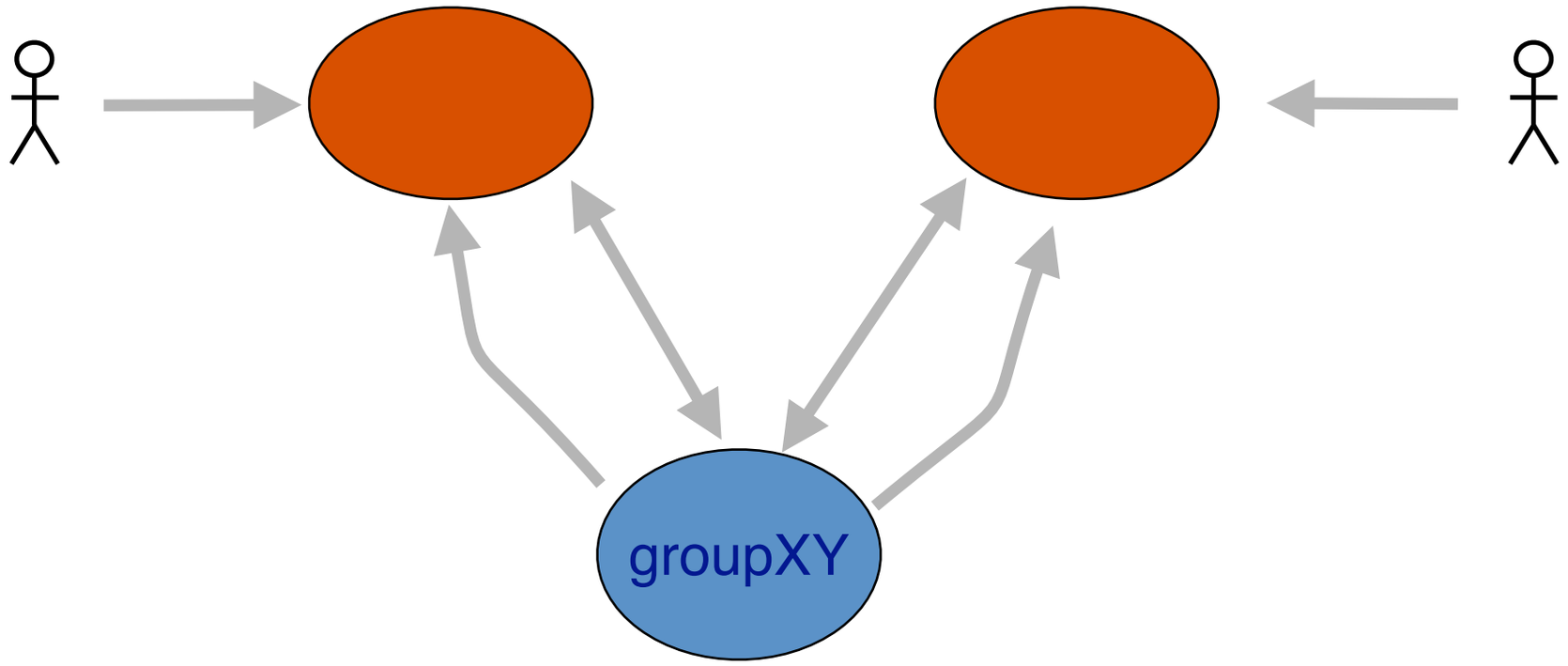
Remove dead/redundant code

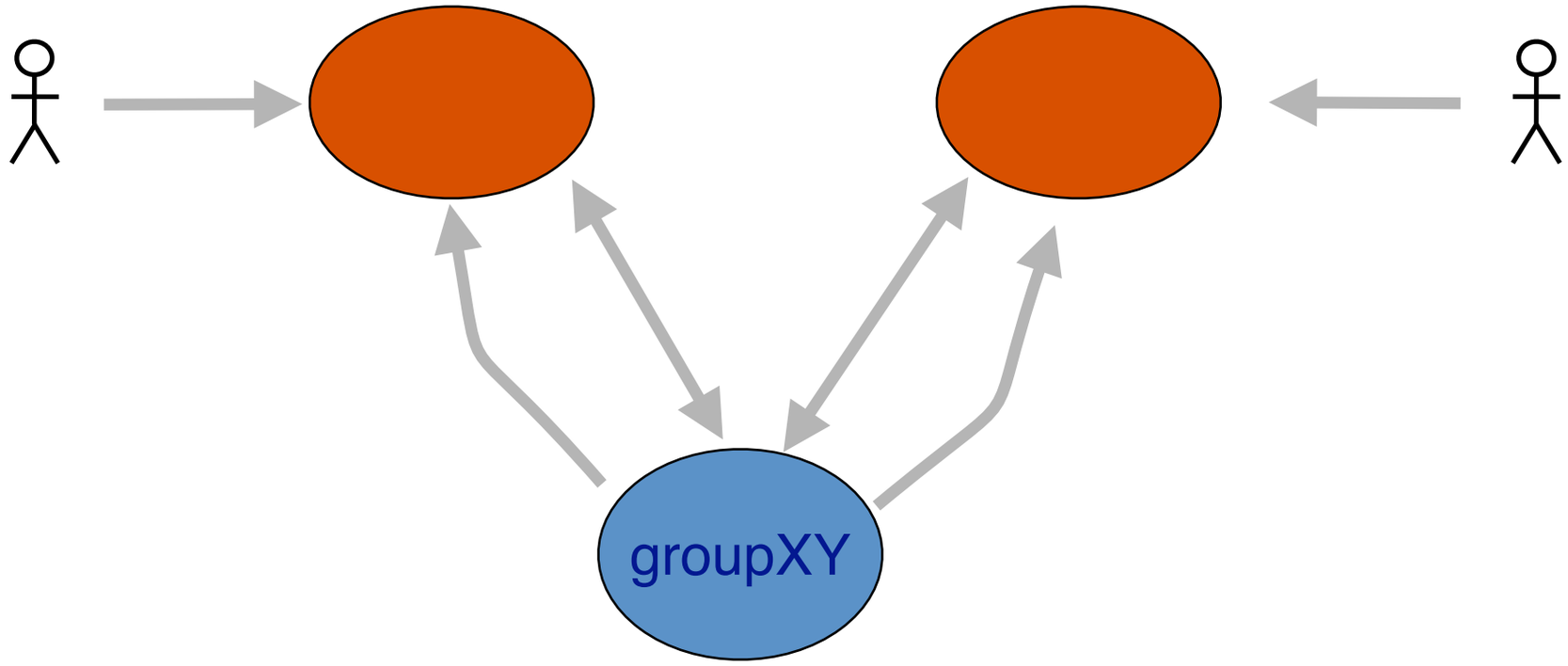
Git clarification

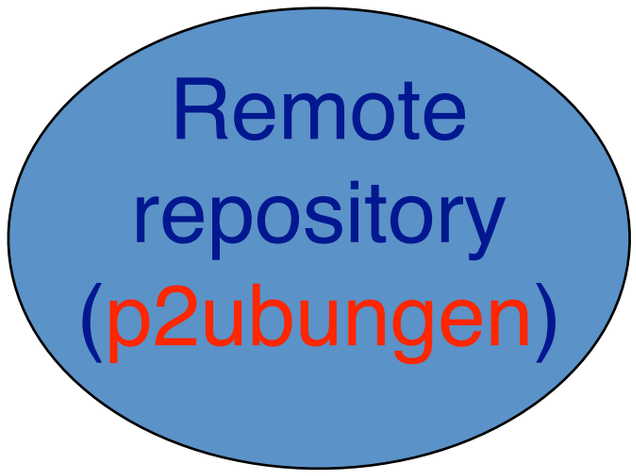

Bob
commit
local repository
Carol
commit
local repository
push/
pull
clone
clone
Remote repository (p2ubungen)

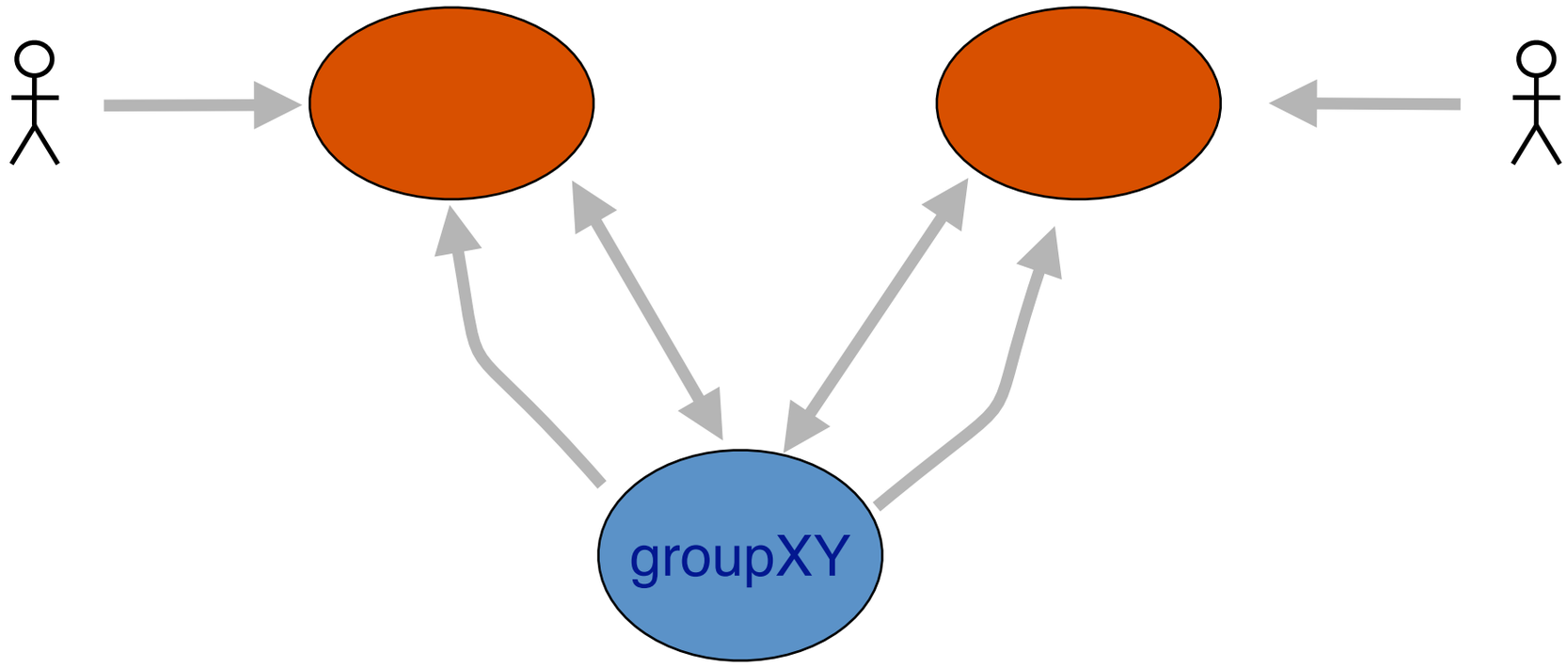

Bob
commit
local repository
Carol
commit
local repository
push/ pull
clone
clone
Remote repository (p2ubungen)

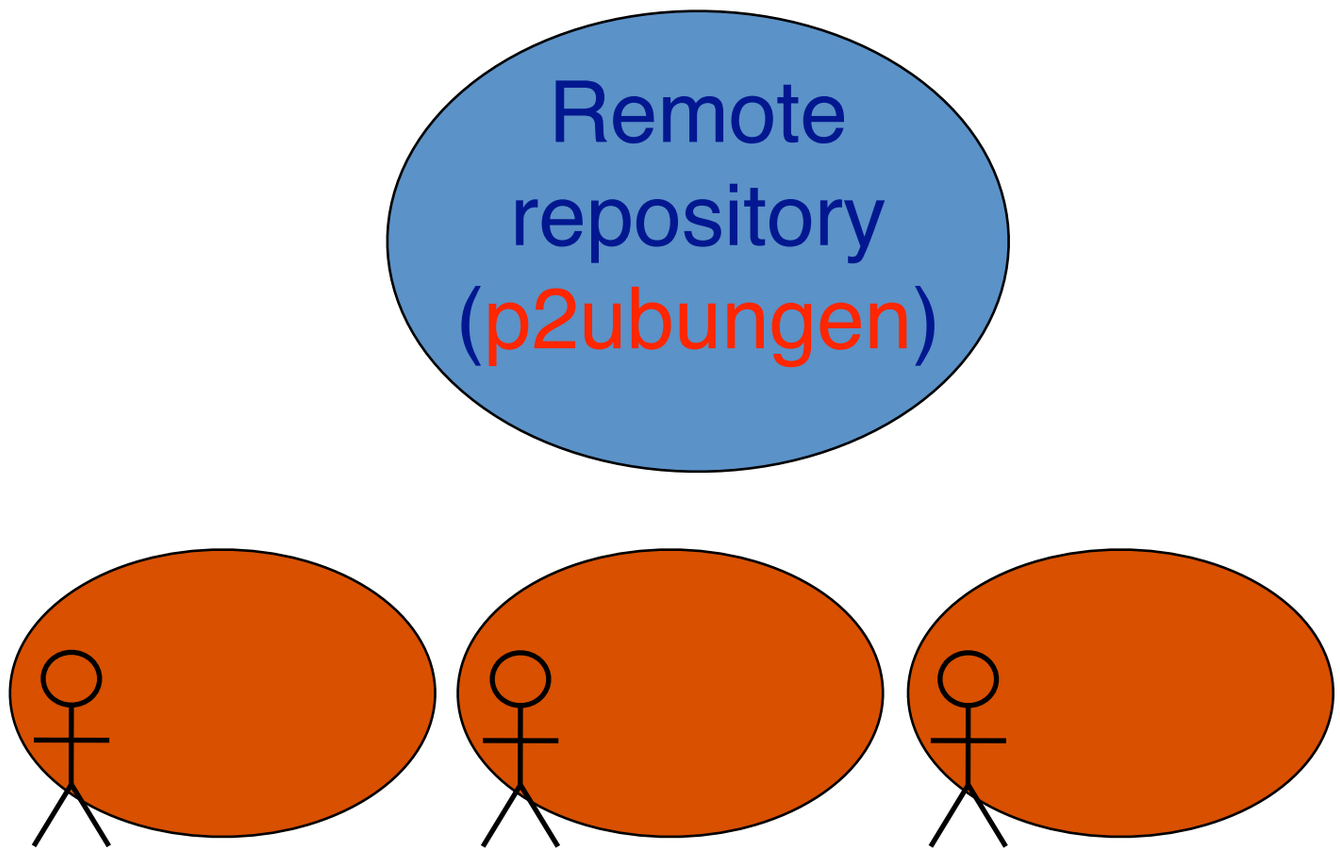

Bob
commit
local repository
Carol
commit
local repository
push/
pull
clone
clone
Remote repository (groupXY)

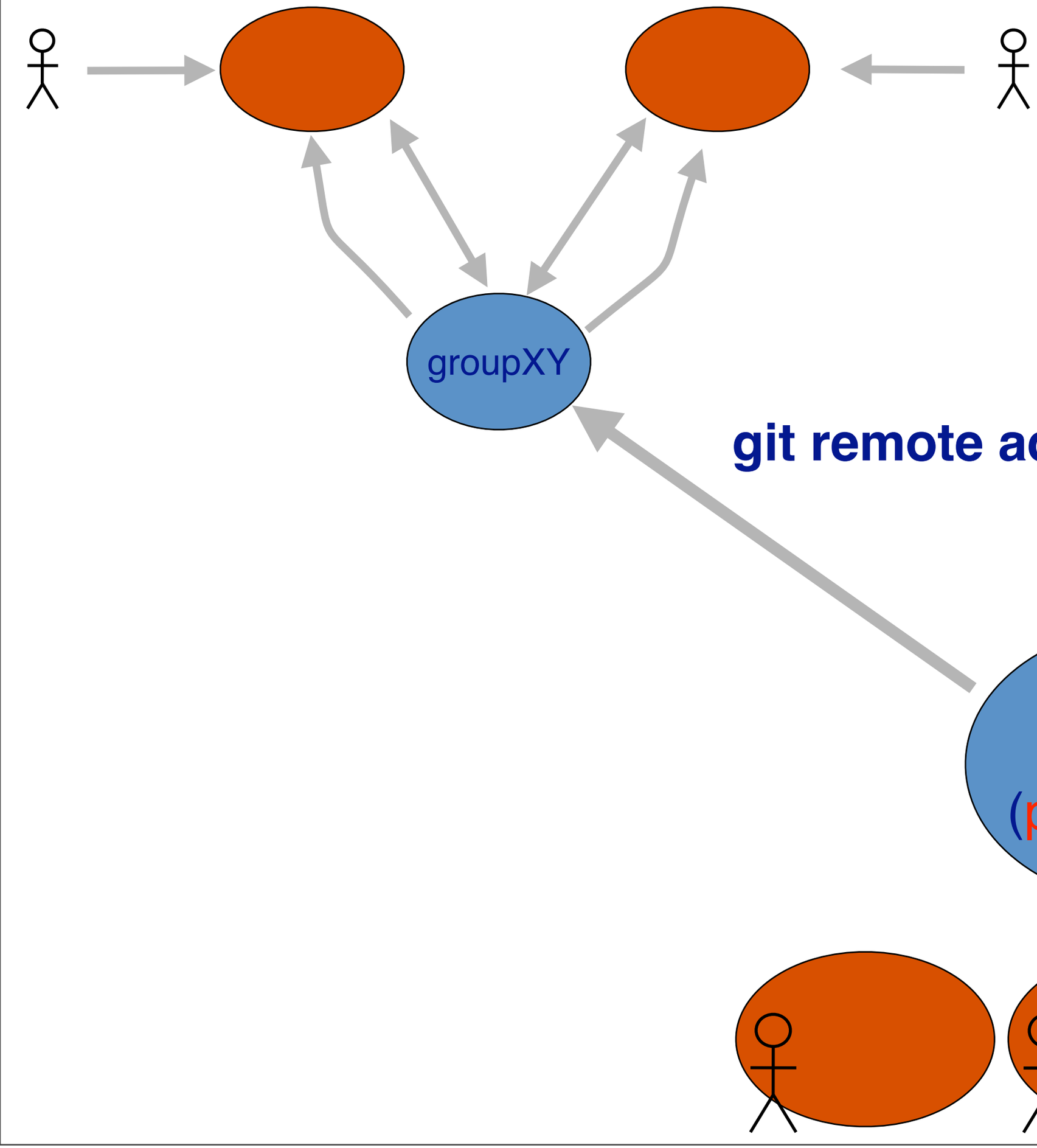

groupXY


groupXY

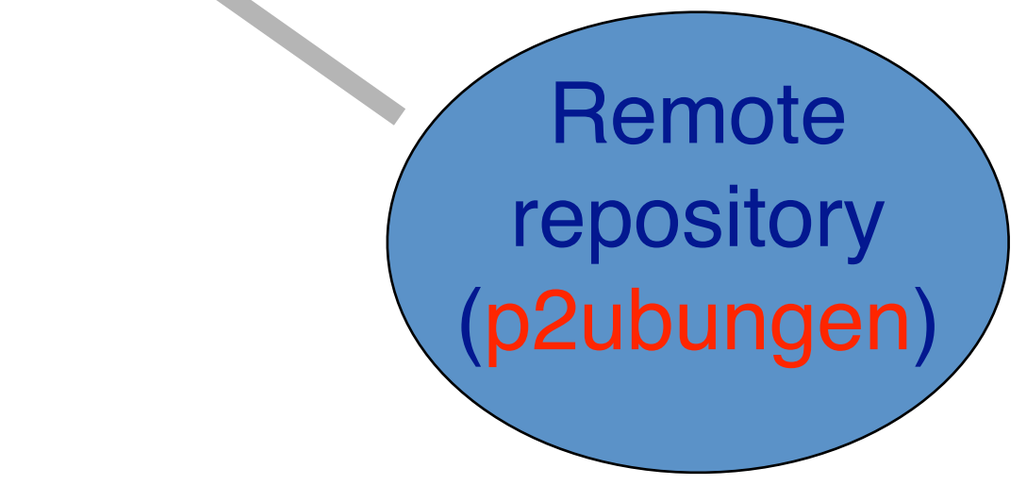

Remote
repository
(p2ubungen)

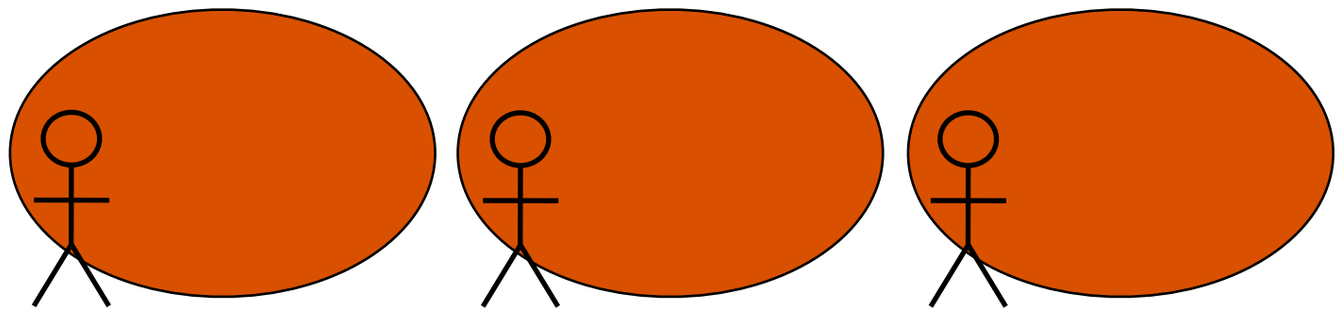

groupXY

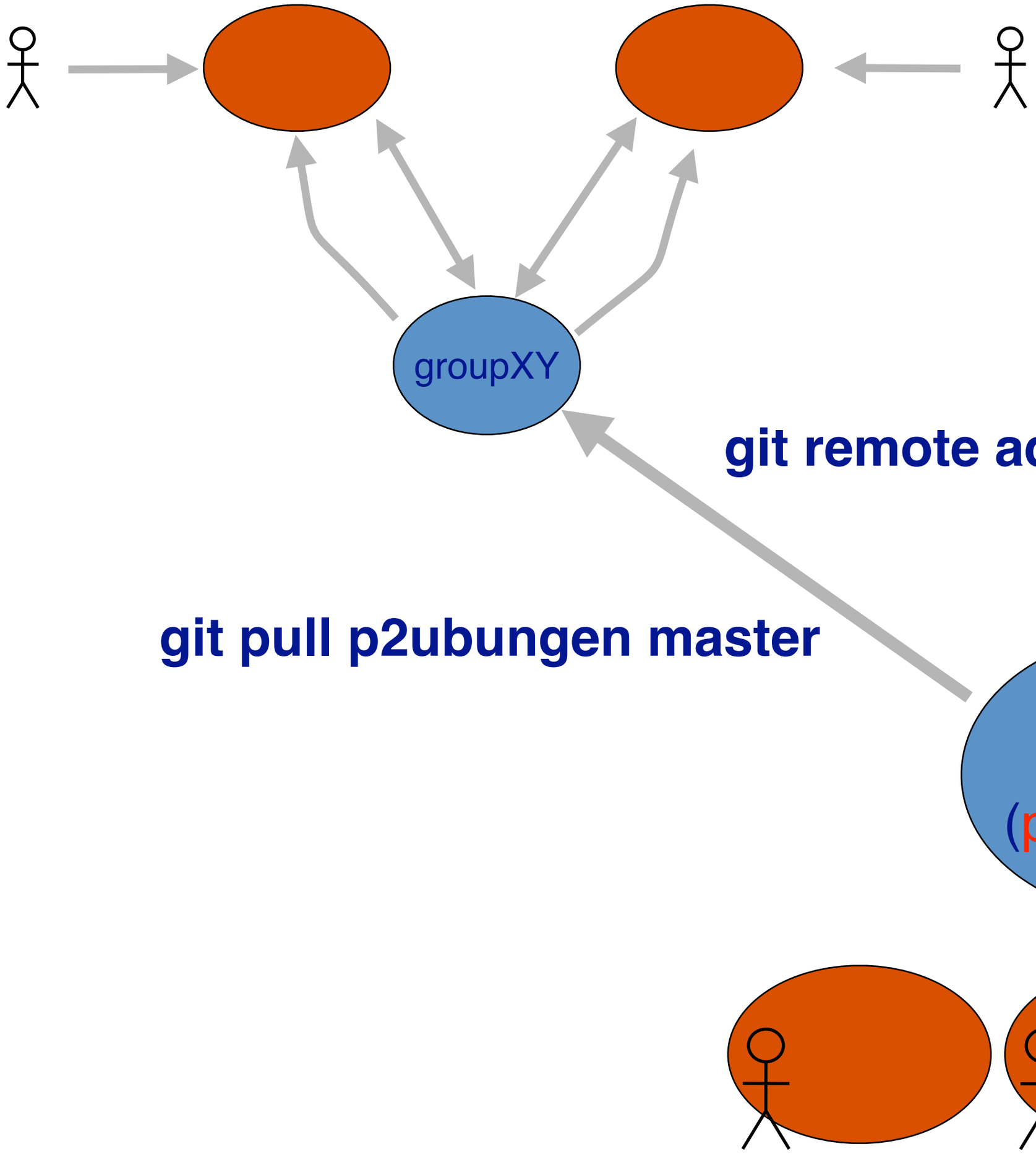

Remote
repository
(p2ubungen)



git remote add p2ubungen ...

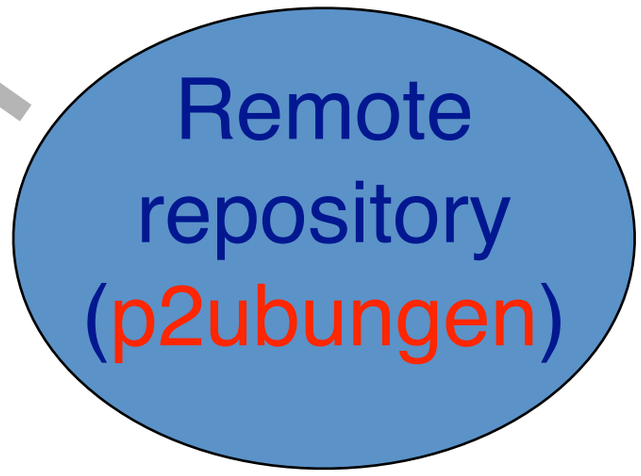


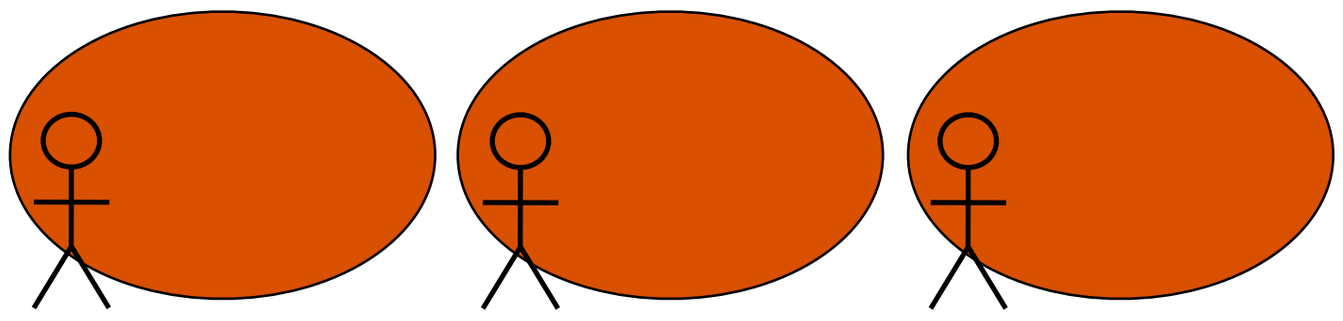


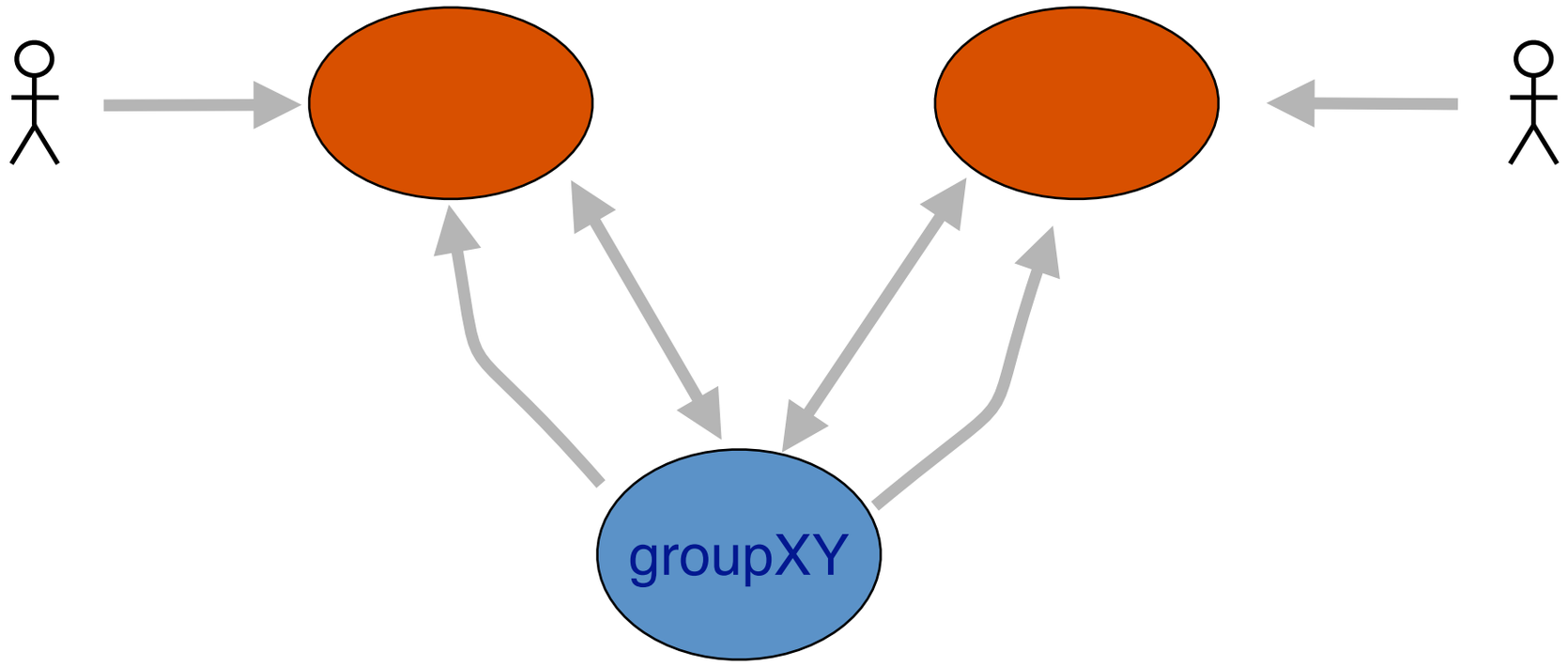


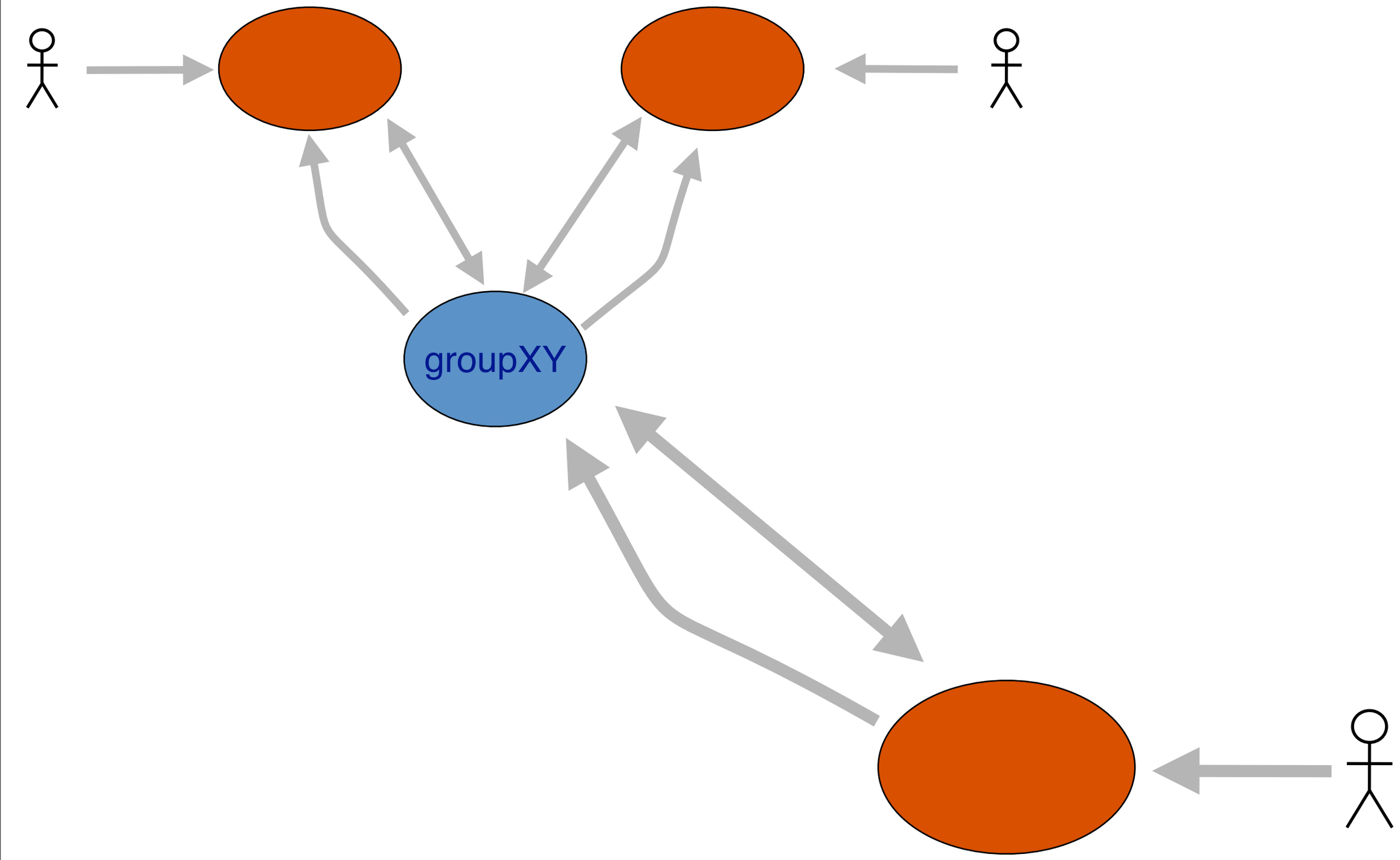
git remote add p2ubungen ...

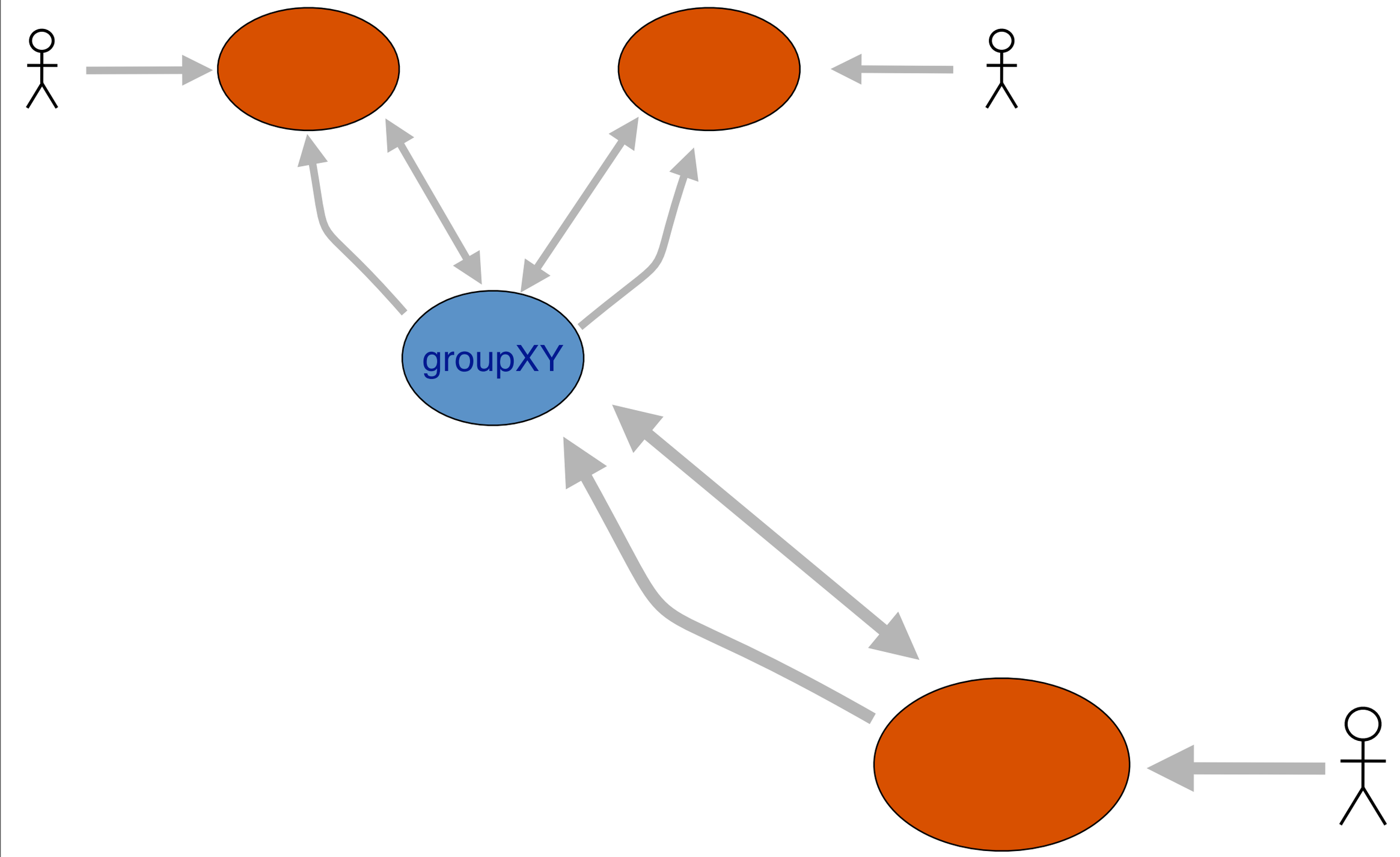
git pull p2ubungen master






groupXY


groupXY



do not commit after the deadline;
it leads to merge conflicts

JExample

In JUnit tests are independent

In JExample tests can depend on other tests

JExample != JUnit

```
import ch.unibe.jexample.JExample;
import ch.unibe.jexample.Given;

@RunWith(JExample.class)
public class SimpleGameTest {
    private Player jack;
    private Player jill;

    ....
}
```

```
@Test
public Game newGame() {
    jack = new Player("Jack");
    jill = new Player("Jill");
    Player[] args = { jack, jill };
    Game game = new Game(12, args);
    game.setSquareToLadder(2, 4);
    assertTrue(game.notOver());
    assertEquals(1, jack.position());
    assertEquals(1, jill.position());
     assertEquals(jack,
          game.currentPlayer());
    return game;
}
```

```
@Test
public Game newGame() {
    jack = new Player("Jack");
    jill = new Player("Jill");
    Player[] args = { jack, jill };
    Game game = new Game(12, args);
    game.setSquareToLadder(2, 4);
    assertTrue(game.notOver());
    assertEquals(1, jack.position());
    assertEquals(1, jill.position());
     assertEquals(jack,
         game.currentPlayer());
    return game;
}
```

```
@Given("newGame")
public Game initialStrings(Game game) {
    assertEquals("Jack", jack.toString());
    assertEquals("Jill", jill.toString());
    assertEquals("[1<Jack><Jill>]",
            game.firstSquare().toString());
    assertEquals("[2->6]",
            game.getSquare(2).toString());
    assertEquals("[5<-11]",
            game.getSquare(11).toString());
    return game;
}
```

```
@Given("initialStrings")
 public Game move1jack(Game game) {
    game.movePlayer(4);
    assertTrue(game.notOver());
    assertEquals(5, jack.position());
    assertEquals(1, jill.position());
    assertEquals(jill, game.currentPlayer());
    return game;
 }
```

What about JUnit?

```
public class SimpleGameTest {
      private Player jack;
      private Player jill;
      private Game game;

          ....
}
```

```
@Before
public void setUp() {
    jack = new Player("Jack");
    jill = new Player("Jill");
    Player[] args = { jack, jill };
    game = new Game(12, args);
    game.setSquareToLadder(2, 4);
}
```

```
@Test
public void testNewGame() {
    assertTrue(game.notOver());
    assertEquals(1, jack.position());
    assertEquals(1, jill.position());
    assertEquals(jack,
        game.currentPlayer());
}
```

```
@Test
public void testInitialStrings() {
    assertEquals("Jack", jack.toString());
    assertEquals("Jill", jill.toString());
    assertEquals("[1<Jack><Jill>]",
            game.firstSquare().toString());
    assertEquals("[2->6]",
            game.getSquare(2).toString());
    assertEquals("[5<-11]",
            game.getSquare(11).toString());
}
```

```
@Test
public void move1jack() {
    game.movePlayer(4);
    assertTrue(game.notOver());
    assertEquals(5, jack.position());
    assertEquals(1, jill.position());
    assertEquals(jill,
        game.currentPlayer());
}
```

http://scg.unibe.ch/research/jexample

http://t.co/N8Fh26y8vZ