**## 🔧 SSL Configuration Options**

**### \*\*Option 1: Disable SSL Verification (Quick Fix for Testing)\*\***

Modify your PersonalJenkinsBot.java to bypass SSL verification:

🛠️ Using tool: fs\_write (trusted)

⋮

● Path: /Users/kombaths/PersonalJenkinsBotNoSSL.java

+ 1: import java.io.\*;

+ 2: import java.net.http.\*;

+ 3: import java.net.URI;

+ 4: import java.util.\*;

+ 5: import java.util.regex.Pattern;

+ 6: import java.util.regex.Matcher;

+ 7: import java.util.Base64;

+ 8: import java.nio.file.Files;

+ 9: import java.nio.file.Paths;

+ 10: import javax.net.ssl.\*;

+ 11: import java.security.cert.X509Certificate;

+ 12:

+ 13: /\*\*

+ 14: \* Personal Jenkins Bot - SSL Disabled Version

+ 15: \* For testing with self-signed certificates

+ 16: \*/

+ 17: public class PersonalJenkinsBotNoSSL {

+ 18:

+ 19: private String jenkinsUrl;

+ 20: private String username;

+ 21: private String apiToken;

+ 22: private final HttpClient httpClient;

+ 23: private final Map<String, String> jobMappings;

+ 24:

+ 25: public PersonalJenkinsBotNoSSL() {

+ 26: this.jobMappings = new HashMap<>();

+ 27: this.httpClient = createInsecureHttpClient(); // SSL disabled

+ 28: loadConfiguration();

+ 29: }

+ 30:

+ 31: /\*\*

+ 32: \* Create HTTP client that accepts all SSL certificates

+ 33: \* WARNING: Only use for testing with trusted servers!

+ 34: \*/

+ 35: private HttpClient createInsecureHttpClient() {

+ 36: try {

+ 37: // Create trust manager that accepts all certificates

+ 38: TrustManager[] trustAllCerts = new TrustManager[] {

+ 39: new X509TrustManager() {

+ 40: public X509Certificate[] getAcceptedIssuers() { return null; }

+ 41: public void checkClientTrusted(X509Certificate[] certs, String authType) { }

+ 42: public void checkServerTrusted(X509Certificate[] certs, String authType) { }

+ 43: }

+ 44: };

+ 45:

+ 46: // Install the all-trusting trust manager

+ 47: SSLContext sslContext = SSLContext.getInstance("SSL");

+ 48: sslContext.init(null, trustAllCerts, new java.security.SecureRandom());

+ 49:

+ 50: // Create HTTP client with custom SSL context

+ 51: return HttpClient.newBuilder()

+ 52: .sslContext(sslContext)

+ 53: .build();

+ 54:

+ 55: } catch (Exception e) {

+ 56: System.err.println("⚠️ Could not create insecure HTTP client: " + e.getMessage());

+ 57: return HttpClient.newHttpClient(); // Fallback to default

+ 58: }

+ 59: }

+ 60:

+ 61: // Rest of the methods remain the same as PersonalJenkinsBot

+ 62: private void loadConfiguration() {

+ 63: Properties props = new Properties();

+ 64: String configFile = "jenkins-config.properties";

+ 65:

+ 66: try {

+ 67: if (Files.exists(Paths.get(configFile))) {

+ 68: props.load(new FileInputStream(configFile));

+ 69: System.out.println("✅ Loaded configuration from " + configFile);

+ 70: } else {

+ 71: System.out.println("⚠️ Config file not found, using defaults");

+ 72: createDefaultConfig();

+ 73: return;

+ 74: }

+ 75: } catch (IOException e) {

+ 76: System.out.println("⚠️ Error loading config: " + e.getMessage());

+ 77: createDefaultConfig();

+ 78: return;

+ 79: }

+ 80:

+ 81: this.jenkinsUrl = props.getProperty("jenkins.url", "https://ci.sms.us.caas.oneadp.com");

+ 82: this.username = props.getProperty("jenkins.username", "");

+ 83: this.apiToken = props.getProperty("jenkins.token", "");

+ 84:

+ 85: if (jenkinsUrl.endsWith("/")) {

+ 86: jenkinsUrl = jenkinsUrl.substring(0, jenkinsUrl.length() - 1);

+ 87: }

+ 88:

+ 89: for (String key : props.stringPropertyNames()) {

+ 90: if (key.startsWith("job.")) {

+ 91: String alias = key.substring(4);

+ 92: String jobName = props.getProperty(key);

+ 93: jobMappings.put(alias, jobName);

+ 94: }

+ 95: }

+ 96:

+ 97: System.out.println("📋 Loaded " + jobMappings.size() + " job mappings");

+ 98: System.out.println("⚠️ SSL verification disabled for testing");

+ 99: }

+ 100:

+ 101: private void createDefaultConfig() {

+ 102: this.jenkinsUrl = "https://ci.sms.us.caas.oneadp.com";

+ 103: this.username = "your-username";

+ 104: this.apiToken = "your-api-token";

+ 105:

+ 106: jobMappings.put("olp-2", "OLP-2");

+ 107: jobMappings.put("olp2", "OLP-2");

+ 108: jobMappings.put("qa-host", "QA-Host-File-Update");

+ 109: jobMappings.put("tests", "Automated-Tests");

+ 110: }

+ 111:

+ 112: public String processPrompt(String prompt) {

+ 113: if (username.isEmpty() || apiToken.isEmpty()) {

+ 114: return "❌ Please configure your Jenkins credentials in jenkins-config.properties";

+ 115: }

+ 116:

+ 117: String lowerPrompt = prompt.toLowerCase().trim();

+ 118:

+ 119: if (containsAny(lowerPrompt, "trigger", "run", "start", "execute", "please", "kick off")) {

+ 120: String jobName = extractJobName(lowerPrompt);

+ 121: if (jobName != null) {

+ 122: return triggerJob(jobName);

+ 123: } else {

+ 124: return "🤔 Which job would you like to trigger? Available: " + String.join(", ", new HashSet<>(jobMappings.values()));

+ 125: }

+ 126: }

+ 127:

+ 128: if (containsAny(lowerPrompt, "status", "check", "how is", "what's the")) {

+ 129: String jobName = extractJobName(lowerPrompt);

+ 130: if (jobName != null) {

+ 131: return checkJobStatus(jobName);

+ 132: } else {

+ 133: return "🤔 Which job status would you like to check?";

+ 134: }

+ 135: }

+ 136:

+ 137: if ((lowerPrompt.contains("list") || lowerPrompt.contains("show")) && lowerPrompt.contains("jobs")) {

+ 138: return listJobs();

+ 139: }

+ 140:

+ 141: if (lowerPrompt.contains("help") || lowerPrompt.equals("?")) {

+ 142: return getHelp();

+ 143: }

+ 144:

+ 145: return "🤔 I didn't understand that. Try:\n• 'trigger OLP-2'\n• 'status of QA host'\n• 'list jobs'\n• 'help'";

+ 146: }

+ 147:

+ 148: private boolean containsAny(String text, String... words) {

+ 149: for (String word : words) {

+ 150: if (text.contains(word)) {

+ 151: return true;

+ 152: }

+ 153: }

+ 154: return false;

+ 155: }

+ 156:

+ 157: private String extractJobName(String prompt) {

+ 158: for (Map.Entry<String, String> entry : jobMappings.entrySet()) {

+ 159: if (prompt.contains(entry.getKey())) {

+ 160: return entry.getValue();

+ 161: }

+ 162: }

+ 163:

+ 164: for (String jobName : new HashSet<>(jobMappings.values())) {

+ 165: if (prompt.toLowerCase().contains(jobName.toLowerCase())) {

+ 166: return jobName;

+ 167: }

+ 168: }

+ 169:

+ 170: return null;

+ 171: }

+ 172:

+ 173: public String triggerJob(String jobName) {

+ 174: try {

+ 175: System.out.println("🔄 Triggering job: " + jobName + " (SSL verification disabled)");

+ 176:

+ 177: String url = String.format("%s/job/%s/build", jenkinsUrl, jobName);

+ 178:

+ 179: HttpRequest request = HttpRequest.newBuilder()

+ 180: .uri(URI.create(url))

+ 181: .header("Authorization", getAuthHeader())

+ 182: .POST(HttpRequest.BodyPublishers.noBody())

+ 183: .timeout(java.time.Duration.ofSeconds(30))

+ 184: .build();

+ 185:

+ 186: HttpResponse<String> response = httpClient.send(request, HttpResponse.BodyHandlers.ofString());

+ 187:

+ 188: if (response.statusCode() == 201) {

+ 189: String queueUrl = response.headers().firstValue("Location").orElse("");

+ 190: return String.format("✅ Job '%s' triggered successfully!\n🔗 Queue: %s\n📊 Check status with: 'status %s'",

+ 191: jobName, queueUrl, jobName);

+ 192: } else if (response.statusCode() == 404) {

+ 193: return String.format("❌ Job '%s' not found. Check the job name or your permissions.", jobName);

+ 194: } else if (response.statusCode() == 401) {

+ 195: return "❌ Authentication failed. Check your username and API token.";

+ 196: } else {

+ 197: return String.format("❌ Failed to trigger job '%s'. HTTP Status: %d", jobName, response.statusCode());

+ 198: }

+ 199:

+ 200: } catch (java.net.ConnectException e) {

+ 201: return "❌ Cannot connect to Jenkins. Check the URL: " + jenkinsUrl;

+ 202: } catch (Exception e) {

+ 203: return "❌ Error: " + e.getMessage();

+ 204: }

+ 205: }

+ 206:

+ 207: public String checkJobStatus(String jobName) {

+ 208: try {

+ 209: String url = String.format("%s/job/%s/lastBuild/api/json", jenkinsUrl, jobName);

+ 210:

+ 211: HttpRequest request = HttpRequest.newBuilder()

+ 212: .uri(URI.create(url))

+ 213: .header("Authorization", getAuthHeader())

+ 214: .GET()

+ 215: .timeout(java.time.Duration.ofSeconds(30))

+ 216: .build();

+ 217:

+ 218: HttpResponse<String> response = httpClient.send(request, HttpResponse.BodyHandlers.ofString());

+ 219:

+ 220: if (response.statusCode() == 200) {

+ 221: String body = response.body();

+ 222: String status = extractJsonValue(body, "result");

+ 223: String buildNumber = extractJsonValue(body, "number");

+ 224: String duration = extractJsonValue(body, "duration");

+ 225:

+ 226: String statusEmoji = getStatusEmoji(status);

+ 227: String durationText = duration != null && !duration.equals("0") ?

+ 228: " (took " + (Long.parseLong(duration) / 1000) + "s)" : "";

+ 229:

+ 230: return String.format("📊 Job '%s' Status:\n%s Build #%s: %s%s\n🔗 %s/job/%s/%s/",

+ 231: jobName, statusEmoji, buildNumber,

+ 232: status != null ? status : "RUNNING", durationText,

+ 233: jenkinsUrl, jobName, buildNumber);

+ 234: } else if (response.statusCode() == 404) {

+ 235: return String.format("❌ Job '%s' not found or no builds yet.", jobName);

+ 236: } else {

+ 237: return String.format("❌ Could not get status for job '%s'. Status: %d", jobName, response.statusCode());

+ 238: }

+ 239:

+ 240: } catch (Exception e) {

+ 241: return "❌ Error checking status: " + e.getMessage();

+ 242: }

+ 243: }

+ 244:

+ 245: private String getStatusEmoji(String status) {

+ 246: if (status == null) return "🔄";

+ 247: switch (status.toUpperCase()) {

+ 248: case "SUCCESS": return "✅";

+ 249: case "FAILURE": return "❌";

+ 250: case "UNSTABLE": return "⚠️";

+ 251: case "ABORTED": return "🛑";

+ 252: default: return "❓";

+ 253: }

+ 254: }

+ 255:

+ 256: public String listJobs() {

+ 257: StringBuilder sb = new StringBuilder();

+ 258: sb.append("📋 Available Jobs:\n");

+ 259: sb.append("==================\n");

+ 260:

+ 261: Set<String> uniqueJobs = new HashSet<>(jobMappings.values());

+ 262: for (String job : uniqueJobs) {

+ 263: sb.append("• ").append(job);

+ 264:

+ 265: List<String> aliases = new ArrayList<>();

+ 266: for (Map.Entry<String, String> entry : jobMappings.entrySet()) {

+ 267: if (entry.getValue().equals(job)) {

+ 268: aliases.add(entry.getKey());

+ 269: }

+ 270: }

+ 271: if (!aliases.isEmpty()) {

+ 272: sb.append(" (aliases: ").append(String.join(", ", aliases)).append(")");

+ 273: }

+ 274: sb.append("\n");

+ 275: }

+ 276:

+ 277: sb.append("\n💡 Usage examples:\n");

+ 278: sb.append("• 'trigger OLP-2'\n");

+ 279: sb.append("• 'status of qa host'\n");

+ 280: sb.append("• 'run automated tests'\n");

+ 281:

+ 282: return sb.toString();

+ 283: }

+ 284:

+ 285: public String getHelp() {

+ 286: return """

+ 287: 🤖 Personal Jenkins Bot - SSL Disabled Version

+ 288: ==============================================

+ 289:

+ 290: ⚠️ WARNING: SSL verification is disabled for testing!

+ 291: Only use with trusted Jenkins servers.

+ 292:

+ 293: Natural Language Commands:

+ 294: • "Please trigger OLP-2"

+ 295: • "Run QA host file update"

+ 296: • "What's the status of OLP-2?"

+ 297: • "List all jobs"

+ 298:

+ 299: Direct Commands:

+ 300: • trigger <job> - Trigger a Jenkins job

+ 301: • status <job> - Check job status

+ 302: • list jobs - Show available jobs

+ 303: • help - Show this help

+ 304:

+ 305: Examples:

+ 306: java PersonalJenkinsBotNoSSL "trigger olp-2"

+ 307: java PersonalJenkinsBotNoSSL "status qa host"

+ 308: """;

+ 309: }

+ 310:

+ 311: private String getAuthHeader() {

+ 312: String credentials = username + ":" + apiToken;

+ 313: return "Basic " + Base64.getEncoder().encodeToString(credentials.getBytes());

+ 314: }

+ 315:

+ 316: private String extractJsonValue(String json, String key) {

+ 317: Pattern pattern = Pattern.compile("\"" + key + "\"\\s\*:\\s\*\"?([^,}\"]+)\"?");

+ 318: Matcher matcher = pattern.matcher(json);

+ 319: return matcher.find() ? matcher.group(1) : null;

+ 320: }

+ 321:

+ 322: public static void main(String[] args) {

+ 323: PersonalJenkinsBotNoSSL bot = new PersonalJenkinsBotNoSSL();

+ 324:

+ 325: if (args.length > 0) {

+ 326: String command = String.join(" ", args);

+ 327: String result = bot.processPrompt(command);

+ 328: System.out.println(result);

+ 329: } else {

+ 330: System.out.println("🤖 Personal Jenkins Bot (SSL Disabled)");

+ 331: System.out.println("Usage: java PersonalJenkinsBotNoSSL \"trigger olp-2\"");

+ 332: }

+ 333: }

+ 334: }

⋮

↳ Purpose: Create SSL-disabled version of PersonalJenkinsBot for testing

Creating: /Users/kombaths/PersonalJenkinsBotNoSSL.java

⋮

**● Completed in 0.0s**

> ### **Option 2: Add SSL Configuration to Properties File**

🛠️ Using tool: fs\_write (trusted)

⋮

● Path: /Users/kombaths/jenkins-config-ssl.properties

+ 1: # Jenkins Configuration with SSL Options

+ 2:

+ 3: # Jenkins Connection

+ 4: jenkins.url=https://ci.sms.us.caas.oneadp.com

+ 5: jenkins.username=kombaths

+ 6: jenkins.token=1133fbd358927c134f9d0f0d851c92a0ab

+ 7:

+ 8: # SSL Configuration Options

+ 9: jenkins.ssl.verify=false

+ 10: jenkins.ssl.truststore.path=

+ 11: jenkins.ssl.truststore.password=

+ 12:

+ 13: # Job Mappings

+ 14: job.olp-2=OLP-2

+ 15: job.olp2=OLP-2

+ 16: job.qa-host=QA-Host-File-Update

+ 17: job.host-file=QA-Host-File-Update

+ 18: job.tests=Automated-Tests

+ 19: job.automated-tests=Automated-Tests

+ 20: job.deploy=Production-Deploy

+ 21: job.deployment=Production-Deploy

⋮

↳ Purpose: Update jenkins config with SSL options

Creating: /Users/kombaths/jenkins-config-ssl.properties

⋮

**● Completed in 0.0s**

> ### **Option 3: Java System Properties (Command Line)**

You can also disable SSL verification using Java system properties:

🛠️ Using tool: fs\_write (trusted)

⋮

● Path: /Users/kombaths/run-with-ssl-disabled.sh

+ 1: #!/bin/bash

+ 2:

+ 3: # Run PersonalJenkinsBot with SSL verification disabled

+ 4: # WARNING: Only use for testing with trusted servers!

+ 5:

+ 6: echo "⚠️ Running with SSL verification disabled"

+ 7: echo "Only use this with trusted Jenkins servers!"

+ 8: echo ""

+ 9:

+ 10: java -Dcom.sun.net.ssl.checkRevocation=false \

+ 11: -Dtrust\_all\_cert=true \

+ 12: -Djavax.net.ssl.trustStore="" \

+ 13: -Djavax.net.ssl.trustStorePassword="" \

+ 14: PersonalJenkinsBot "$@"

⋮

↳ Purpose: Create script to run Java with SSL disabled via system properties

Creating: /Users/kombaths/run-with-ssl-disabled.sh

⋮

**● Completed in 0.0s**

🛠️ Using tool: execute\_bash (trusted)

⋮

● I will run the following shell command:

chmod +x /Users/kombaths/run-with-ssl-disabled.sh

⋮

↳ Purpose: Make the SSL disabled run script executable

⋮

**● Completed in 0.18s**

> ## 🚀 Let's Test the SSL-Disabled Version

🛠️ Using tool: execute\_bash (trusted)

⋮

● I will run the following shell command:

cd /Users/kombaths && javac PersonalJenkinsBotNoSSL.java

⋮

↳ Purpose: Compile and test the SSL-disabled version

⋮

**● Completed in 0.429s**

🛠️ Using tool: execute\_bash (trusted)

⋮

● I will run the following shell command:

cd /Users/kombaths && java PersonalJenkinsBotNoSSL "trigger olp-2"

⋮

↳ Purpose: Test the SSL-disabled version with trigger command

✅ Loaded configuration from jenkins-config.properties

📋 Loaded 8 job mappings

⚠️ SSL verification disabled for testing

🔄 Triggering job: OLP-2 (SSL verification disabled)

✅ Job 'OLP-2' triggered successfully!

🔗 Queue: https://ci.sms.us.caas.oneadp.com/queue/item/53247/

📊 Check status with: 'status OLP-2'

⋮

**● Completed in 2.552s**