

IC Flow Platform 用戶手册

Product Name : IFP (IC Flow Platform)

Product Version : V1.3

Release Date : 2024.07.15

Contact : [@李艳青](#) (liyanqing.1987@bytedance.com)

[@景富义](#) (jingfuyi@bytedance.com)

[@节 喜](#) (jiexi@bytedance.com)

[@马 琪](#) (makun.226@bytedance.com)

[@张静文](#) (zhangjingwen.silvia@bytedance.com)

前言	4
1. 简介	5
1.1 IC 流程演化	5
1.2 IC 流程平台的主要作用	5
1.3 IFP 开发思路	6
1.4 IFP 基本概念和控制逻辑	7
1.4.1 任务拆解（从 Block 到 Task）	7
1.4.2 单个任务的行为（从 Task 到 Action）	7
1.4.3 单个行为的属性（从 Action 到 Attribute）	8
1.4.4 IFP 控制逻辑	8
1.5 排版和语法约定	9
2. 快速上手案例	10
2.1 启动工具	10
2.2 调整设置	10
2.3 执行任务	14
2.4 查看报告	15
3. 功能介绍	17
3.1 菜单栏	17
3.1.1 File	17
Save Status File	17
Load Status File	17
Save Config File	17
Load Config File	17
3.1.2 View	17
Config View	18
Main View	18
Zoom In	18
Zoom Out	18
Detail Status Info	18
3.1.3 Control	19
3.1.4 Tool	19
LSF monitor	20
3.1.5 Help	20
About	21
Guidance	21
3.2 功能区	22
3.3 用户设置区域	23
3.3.1 系统设置界面	23
Project settings	23
System settings	24
Advanced settings	25
3.3.2 任务设置区域	26

创建任务	26
调整任务属性	27
3.3.3 依赖关系设置区域	28
设置 Flow 之间的依赖关系	28
设置 Task 之间的依赖关系	30
3.3.4 内置变量设置区域	31
可编辑变量	31
不可编辑变量	31
3.3.5 应用程序接口设置区域	32
3.4 任务监控区域	32
3.4.1 任务层级区域	33
3.4.2 主要工作区域	33
展示信息	33
菜单功能	34
按钮功能	35
表头功能	37
3.4.3 任务状态汇总区域	37
3.4.4 日志信息区域	38
4. 技术支持	40
4.1 变更历史	40
4.2 外部贡献者	41

前言

IC Flow Platform（下文中简称 IFP）是 ByteDance 开源的芯片设计流程平台，主要用于超大规模数字集成电路设计的流程规范管理和数据流转控制。

IFP 是基于项目和用户组的一站式工作平台，能够适配各种不同的设计环境（DV/DFT/SYN/PD/K 库等），通过层次解耦的设计思路拆分方法学和自动化流程。提供集约化的数据管理和展示，大幅提高多项目并行、多用户协作的工作效率。在统一管理的前提下，通过丰富的 API 功能提供高自由度的拓展性，从而收敛日常工作中与主流程相关的辅助工作，满足不同用户的需求。

管理员请参考《IC Flow Platform 管理员手册》配置项目组/用户组的默认设置和 API 功能。

用户请参考《IC Flow Platform 用户手册》在管理员默认配置基础上使用 IFP 进行日常工作。

用户可按照 [章节 2 快速上手案例](#) 试用 IFP，了解推荐的运作逻辑及基本功能。

1. 简介

1.1 IC 流程演化

数字集成电路设计流程演进可以笼统地分为三个阶段。

手工阶段：

集成电路设计流程以手工为主，包括手工创建运行目录并解决环境依赖，手工运行 EDA 工具并检查运行结果，手工收集报告并 release 数据。

这阶段的主要瓶颈是，在集成电路设计规模较大，并且人力资源不够充足的情况下，手工操作效率较低，工作量巨大。

脚本阶段：

通过脚本化，包括但不限于 shell/Makefile/perl/python 等，将设计流程步骤中的手工操作转化为脚本执行，这样不但可以极大地提升运行效率，而且可以将流程步骤固化。

这阶段的主要瓶颈是，在超大规模集成电路设计中，每个流程往往需要多人协作，多个流程之间数据流转也更加紧密，因此对流程运行规范和行为一致性提出了更高的要求。

平台阶段：

通过一个统一的流程平台，在一个公司（或团队）范围内统一流程运行规范，数据集约化管理和展示，可以大幅降低流程运行成本（包括时间成本和交互成本）。

这个阶段的主要制约因素是，大家的操作对象从分立脚本切换到统一平台，有一个一次性的学习成本；部分流程管理员失去了对操作平台功能的直接修改权限会有一些不适应。

1.2 IC 流程平台的主要作用

统一流程运行规范：

超大规模集成电路设计需要多流程参与，每个流程内部也需要多人协同工作，流程平台可以将 IC 流程运行规范通过平台工具的方式固化，确保所有的人采用统一的运行模式和规则，降低交互成本。

同分立的自动化脚本相比，流程平台是多流程唯一入口且无法随意篡改的。

数据集约化管理和展示：

流程平台本身承载数据检查及展示（checklist/summary）和数据管理（release）的作用，同时可以嵌入更长维度的数据保存及检索能力，可以对多项目/多流程/多用户的数据进行集约化管理和展示，对多流程间的数据交互更友好。

降低流程运行门槛：

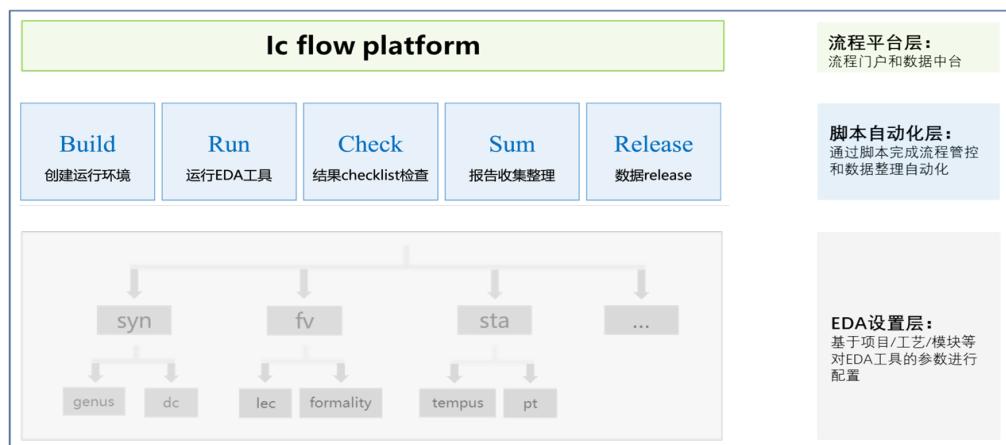
GUI 界面的流程平台简单易操作，可以大幅降低新人的流程运行门槛。同时“脚本自动化层”和“EDA 设置层”通过松耦合的方式嵌入，也不影响用户了解和配置单步自动化的实现方式以及具体的 EDA 设置项。

提升多模块运行效率：

流程平台内嵌基于 multi-thread 或者 LSF / openlava 的多模块并行运行控制，能够在 block / version / flow / vendor / branch / task 不同的层面实现串并行精准管理。

1.3 IFP 开发思路

不同于很多公司将 EDA 配置和流程平台打包的方式（比如 AMD 的 TileBuilder），为了更好地适配各家不同的 EDA 环境，IFP 采用层次解耦的设计思路，将整个 IC 流程组织架构分为三个逻辑层次。



底层是 EDA 设置层，也是绝大多数 project engineer 日常接触到的部分。用户可根据自己所参加的项（如 cpu/gpu/ai/...），项目所使用的工艺节点（16/7/5/...），自己所选用的 EDA 工具（dc_shell/pt_shell/innovus/...），自己所负责的模块（PCIE/DDR/...），来决定如何设置 EDA 工具的配置文件，以保证 EDA 工具能够根据自己的合理配置运行得到期望的结果。

中间是脚本自动化层，也是每个 flow administrator 所维护的部分。针对单个的 block 而言，实际的 IC 流程运行并非单纯运行 EDA 工具，还需要预先准备运行环境（目录结构，input 文件，环境配置等），按照 checklist 检查 EDA 运行结果，从运行目录收集和整理重要报告及数据，结果符合预期的情况下进行数据 release，有时候还有一些后处理的工作要做，而通过脚本，无论是 shell/Makefile 还是 perl/python，都可以大幅降低这部分工作的操作难度。除此之外，多模块/多流程同时运行也可以借助脚本实现并行控制和顺序管控。

顶层是流程平台层，是 IC 用户 EDA 流程的入口。比较知名的 IC 流程平台有 AMD 的 TileBuilder/达索的 Altair FlowTracer/synopsys 的 Lynx 等，一般采用 GUI 界面的方式，支持多 block/version/flow/vendor/branch/task 的并行运行，并集成自动化脚本实现各个流程的自动化运行。

总地来说：

IFP 主要涵盖流程平台层，用来调度和管理用户任务。

自动化脚本仍然由每个 flow 的管理员自行维护和管理，并非 IFP 内嵌的一部分。

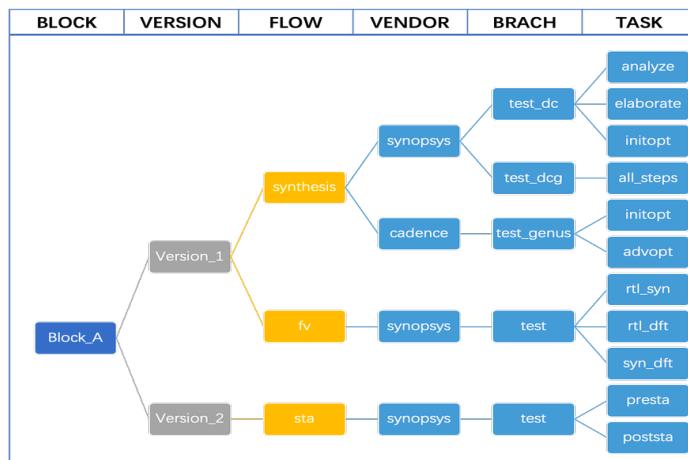
EDA 设置由 flow 的管理员和相关用户自行维护和管理，并非 IFP 内嵌的一部分。

每个流程既可以采用流程平台运行，也可以采用脚本运行，只是流程规范和功能完备性上的区别。

1.4 IFP 基本概念和控制逻辑

1.4.1 任务拆解（从 Block 到 Task）

IFP 的任务拆解逻辑如下，以 mid-end 流程做例子。



Block，即集成电路模块。之所以把 Block 放到任务划分的顶层，是因为一般自顶向下的数字集成电路设计流程，大多按照 Block 来划分任务。

Version，即 Block 不同设计阶段的版本信息。比如 PN85、PN95_v1 等。

Flow，即集成电路设计的流程环节。比如 regression、DFT、synthesis、fv、sta 等，不同公司的划分和命名方式会有些不同。

Vendor，即运行某个流程所用到的哪家 EDA 厂商的工具。比如针对 synthesis 流程，常用的工具有 cadence 的 genus 和 synopsys 的 dc_shell，所以存在 cadence 和 synopsys 两个 Vendor 可选。（不同 Vendor 会影响 checklist/summary 等脚本的选择）。

Branch，用于同一 Vendor 工具的不同模式区分。比如用 synopsys 的 dc_shell 来跑 synthesis 流程，也可能存在 dc 和 dcg 两种不同的运行模式。

Task，每一个运行环节的最小的运行逻辑，即一个任务。比如 dc_shell 跑 synthesis，可以细分为 analyze/elaborate/initopt/advopt/finalopt 等不同的步骤，每个步骤都是一个 Task。

1.4.2 单个任务的行为（从 Task 到 Action）

每个 IC 设计流程的最小任务（task），都可以通过几个逻辑行为来实现，把每个逻辑行为称为一个 Action。

每一个 task 的实现步骤，都可以抽象为如下 5 种 Action。



每一个 Task 可能只用到其中的一种或者几种 Action，也可能用到全部 Action。

1.4.3 单个行为的属性（从 Action 到 Attribute）

对于每个 Action 而言，需要设定一些基本属性（Attribute）以帮助 IFP 确定如何实现这一 Action，其基本的属性如下：

PATH: 在哪个路径下运行这个行为。

COMMAND: 如何运行这个行为（执行的命令）。

RUN_METHOD: 针对任务运行前的一些设置。

也就是说，要完成一个任务（比如 synthesis 的 initopt），都可以通过指定的几个步骤完成（创建环境、运行工具、结果检查...），每个步骤的行为都是“**到指定 PATH 下面，执行 COMMAND，COMMAND 的运行方式为 RUN_METHOD**”。

针对 CHECK 和 SUMMARIZE 这两个 Action，因为会生成报告，还需要查看报告，所以还有如下两个额外项目需要配置。

VIEWER: 是用什么工具或者命令来查看报告，可以带参数。

REPORT_FILE: 报告文件的具体位置。

这样就可以把最小任务（task）的完成，拆解为几个行为（Action），又通过定义每个行为的属性来最终实现任务。

1.4.4 IFP 控制逻辑

IFP 本质上是一个带图形界面的逻辑控制机，输入复杂的用户需求，吐出统一的流程数据。



用户输入任务需求，任务需求包括三方面：

Block/Version/Flow/Vendor/Branch/Task 信息，在用户配置文件（user config file）中定义。

每个任务（Task）的行为（Action）信息，可以在用户配置文件（user config file）中实现，也可以让流程管理员在默认配置文件（default config file）中预先定义。

每个行为（Action）的属性（Attribute）信息，可以在用户配置文件（user config file）中实现，也可以让流程管理员在默认配置文件（default config file）中预先定义。

IFP 主要负责按照用户的初始信息（用户配置文件）和操作指令（GUI 界面的点击操作），并行或者串行运行用户指定的任务，并最终汇总输出结果。

1.5 排版和语法约定

<u>章节 1.5</u>	指可以在对应章节中找到更为详细的使用说明
按钮	指在 IFP 界面上用户可以点击操作的按钮
<i>Setup -> Setting</i>	指 IFP 菜单栏
<code> \${IFP_INSTALL_PATH}</code>	指需要用户根据实际使用场景自行替换的变量或在 IFP 中可以以变量形式调用的参数
<i>auto_import_tasks</i>	指 IFP 中用户可控制的系统选项

2. 快速上手案例

设置环境变量 \${IFP_DEMO_MODE} 为 TRUE，则开启测试案例模式。

2.1 启动工具

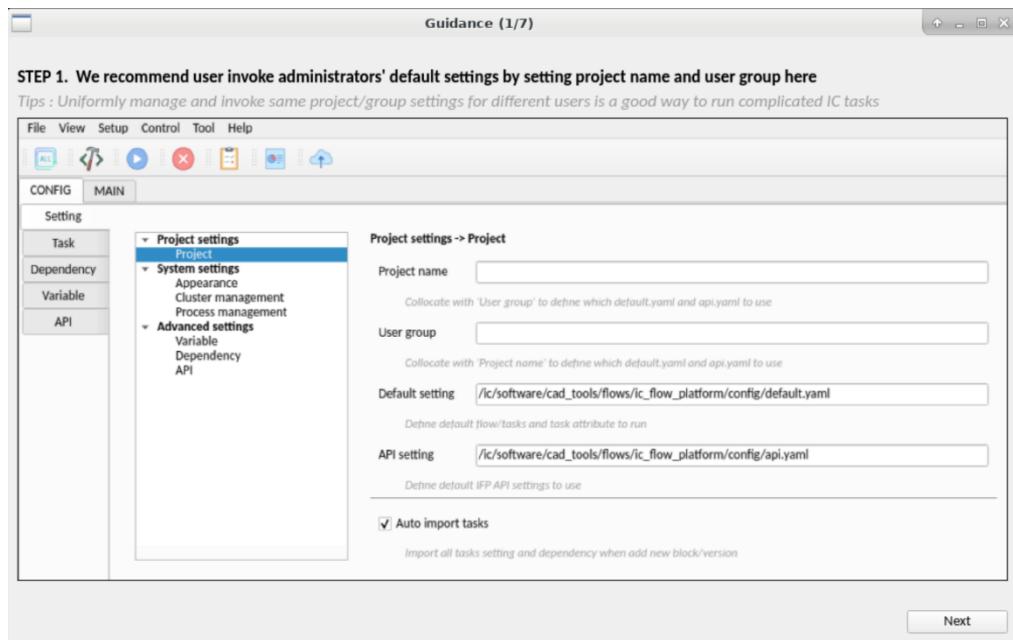
(1) 输入如下命令（案例为 bash 环境），其中 \${IFP_INSTALL_PATH} 为管理员安装 IFP 的路径：

```
export IFP_DEMO_MODE=TRUE
${IFP_INSTALL_PATH}/bin/ifp
```

(2) 将看到如下提示，本案例通过 IFP 的 PRE_CFG (API) 功能自动为用户创建 DV 测试案例工作环境：

```
>>> IFP Demo mode, you can set $IFP_DEMO_MODE=FALSE to exit
[API] : You can execute some script before load ifp.cfg.yaml by API(PRE_CFG) function, such as generate a customized ifp.cfg.yaml for user!
[API] : *Info*: Execute IFP API (PRE_CFG) function to generate demo case database!
[API] : *Info*: USER : jingfuyi
[API] : *Info*: PROJECT : demo
[API] : *Info*: GROUP : dv
[API] : You can execute some script after load ifp.cfg.yaml and before launch GUI by API(PRE_IFP) function, such as a Wrapper to assist user generate EDA environment!
```

(3) 之后将自动加载环境，用户可参考导览界面大致了解图形界面主要功能：

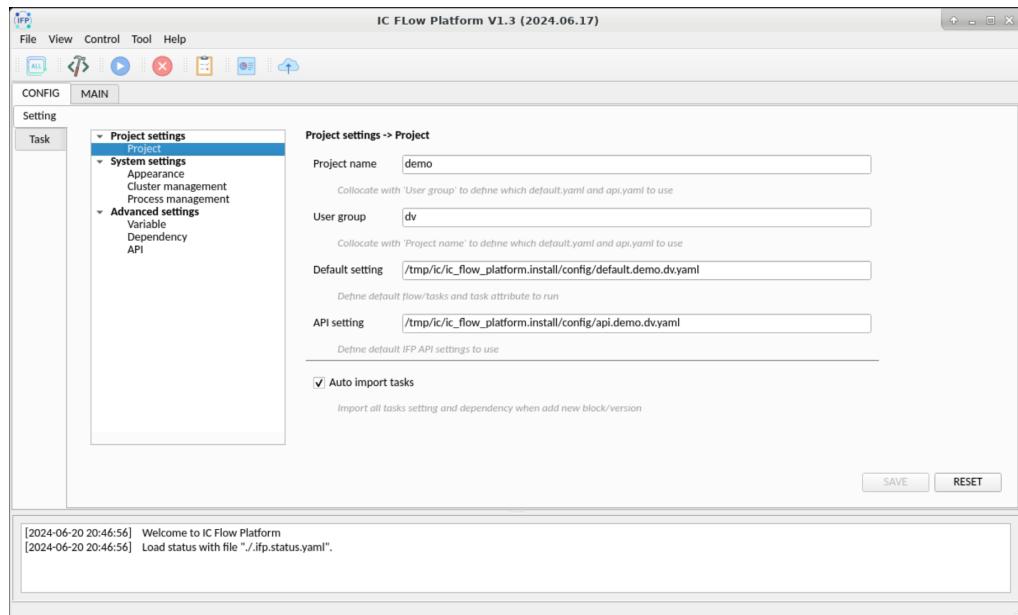


2.2 调整设置

(1) IFP 的 PRE_CFG 功能自动匹配用户的项目组为 demo，用户组为 dv，则调用的默认流程和 API 配置如下，管理员已为用户配置好默认设置，用户无需或仅需微调即可：

```
`${IFP_INSTALL_PATH}/config/default.demo.dv.yaml
`${IFP_INSTALL_PATH}/config/api.demo.dv.yaml
```

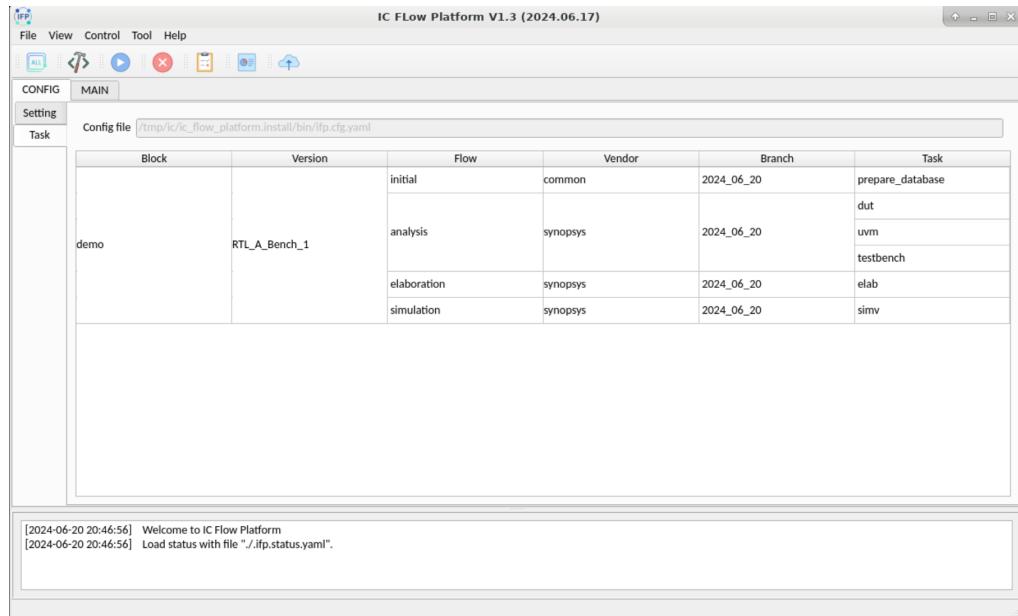
(2) 在 CONFIG- Setting 界面 可以查看系统设置：



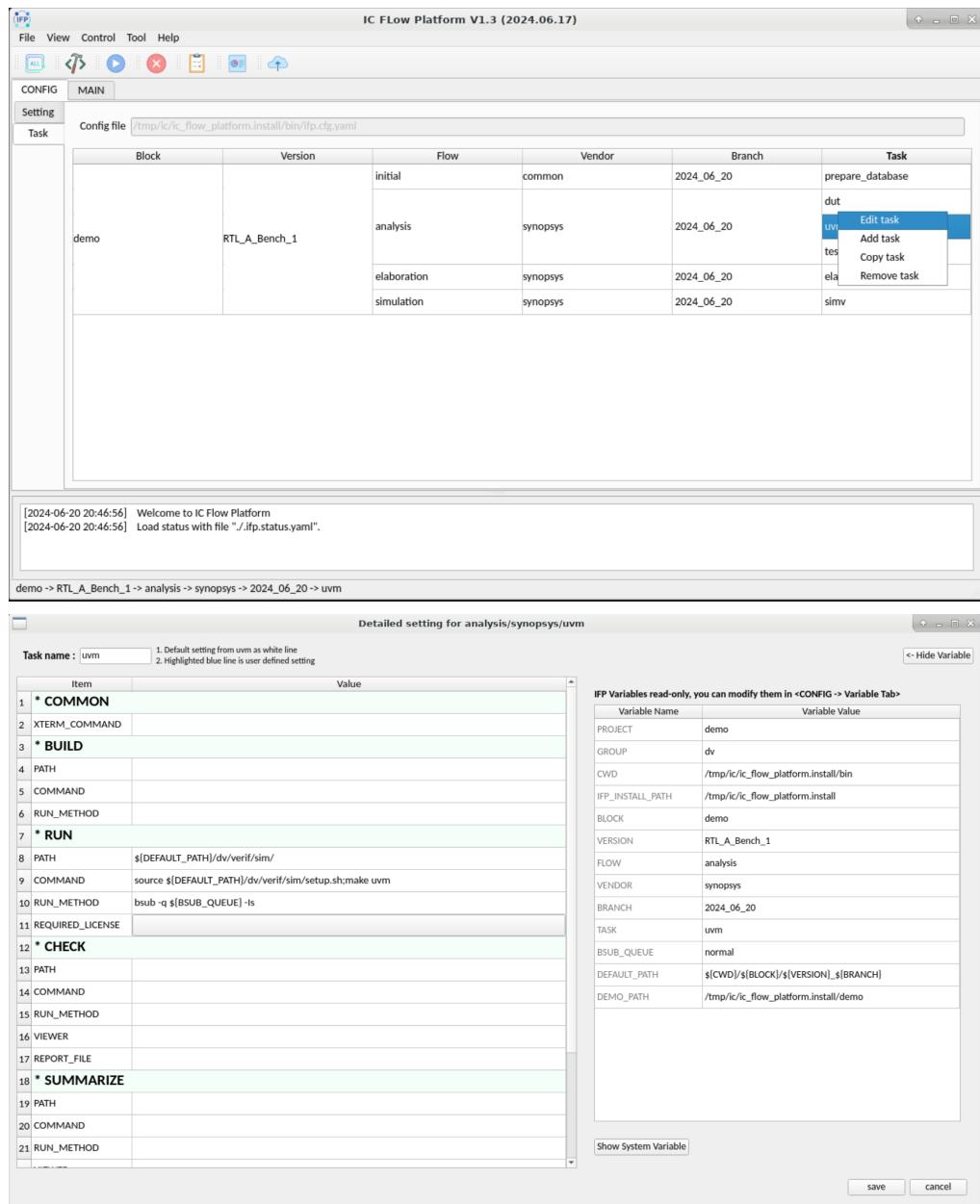
需要注意的是，用户需要调整 *Cluster management* 中的 `$BSUB_QUEUE`，因为本案例采用 LSF 集群的方式分发任务。

另外可以在 *Advanced settings* 中打开高阶功能调整其他参数，例如 *Variable*（变量设置），*Dependency*（依赖关系设置），*API*（应用程序接口设置）。

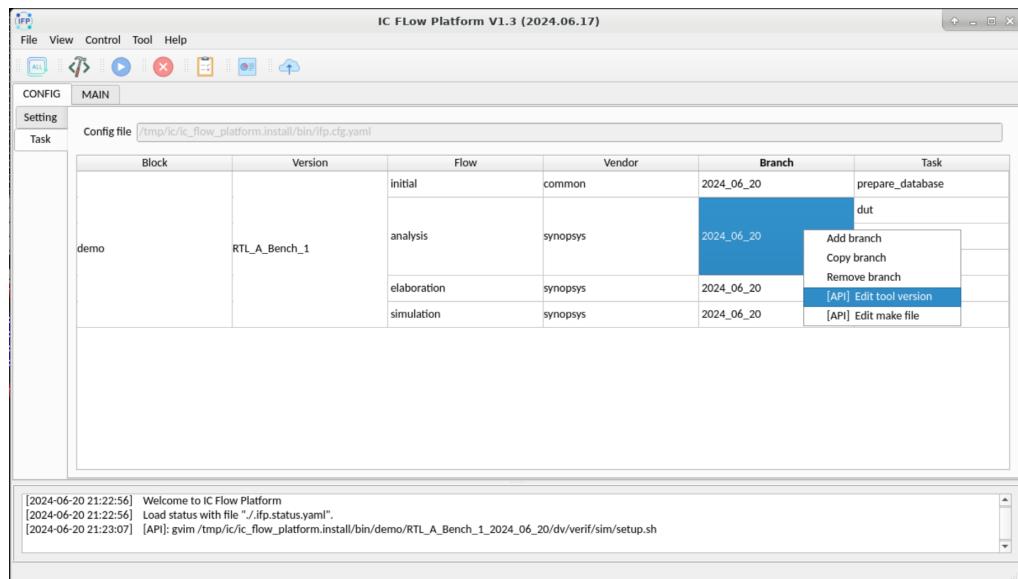
(3) 接下来在 *CONFIG-Task* 界面可以查看任务设置：



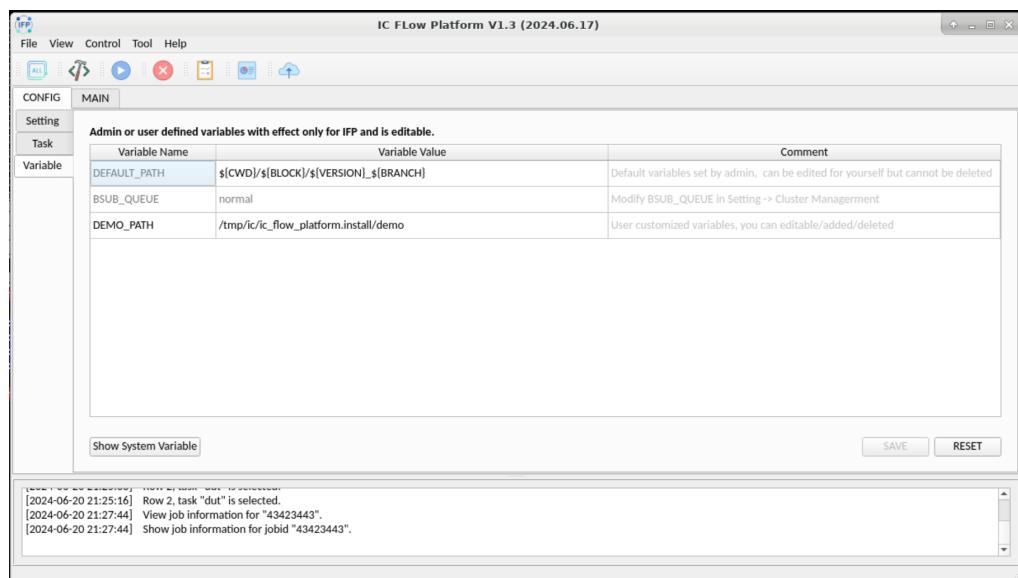
(4) 在 *Task* 栏位 点击右键，弹出菜单中选择 *Edit task*，可以查看管理员为该任务配置的详细参数，用户也可以在此进行微调：



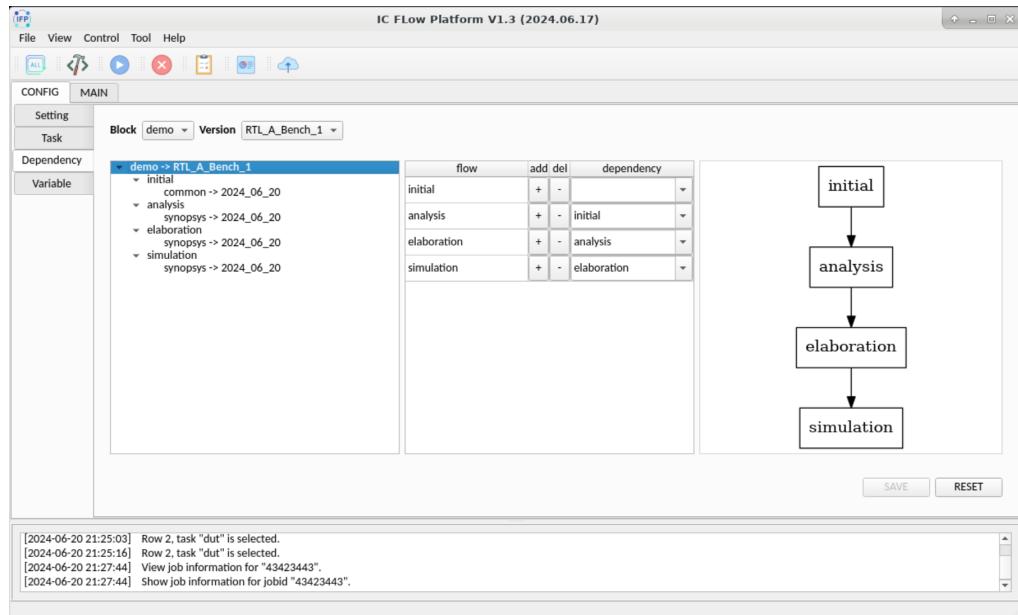
(5) 通过绑定在 *Branch* 栏位 右键菜单中的 API 功能，可以快速访问包含 Tool 版本的文件，请根据实际环境进行更改：



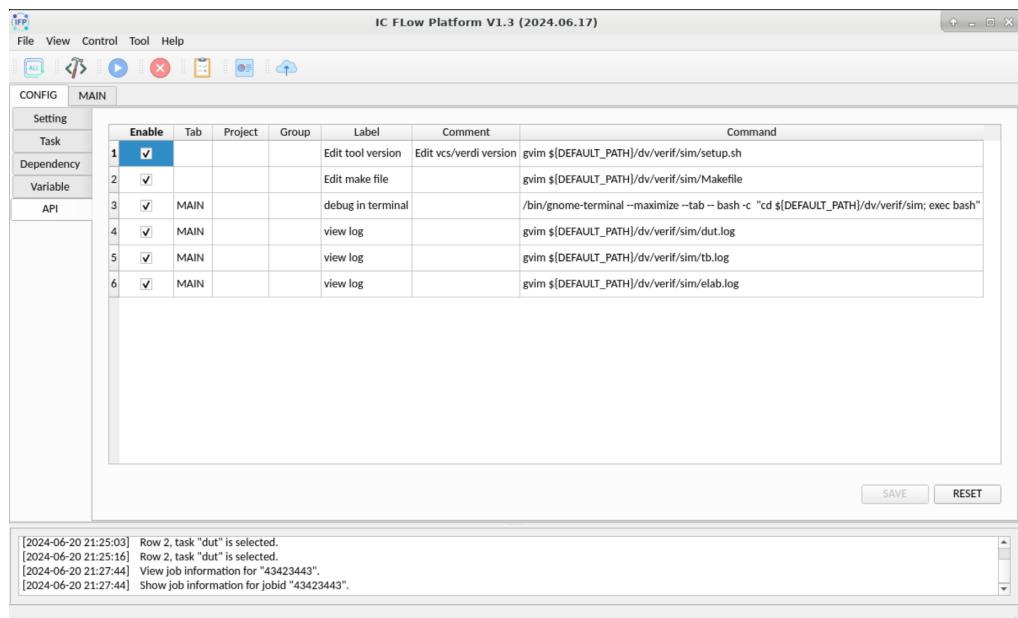
(6) 若用户打开了 *Advanced settings - Variable - Enable user variable interface*, 则可在 *CONFIG-Variable* 界面 查看到设置的 IFP 内置变量:



(7) 若用户打开了 *Advanced settings - Dependency - Enable user dependency interface*, 则可在 *CONFIG-Dependency* 界面 查看到 Flow/Task 间的依赖关系:

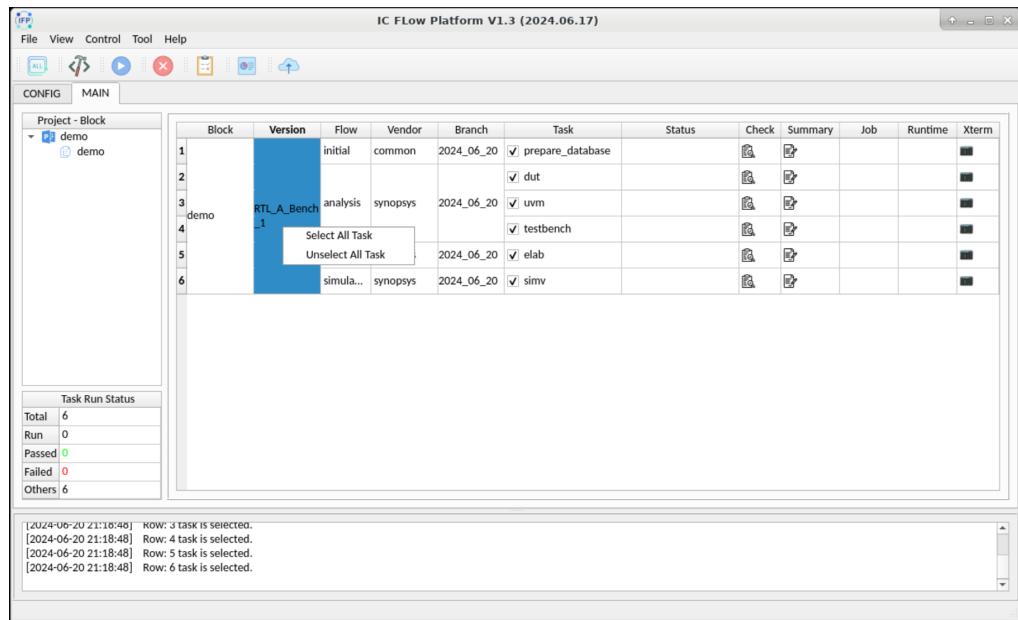


(8) 若用户打开了 *Advanced settings - API - Enable user API interface*, 则可在 *CONFIG-API 界面* 查看到为 Demo case 定制的辅助功能以及这些功能在界面上生效的位置:

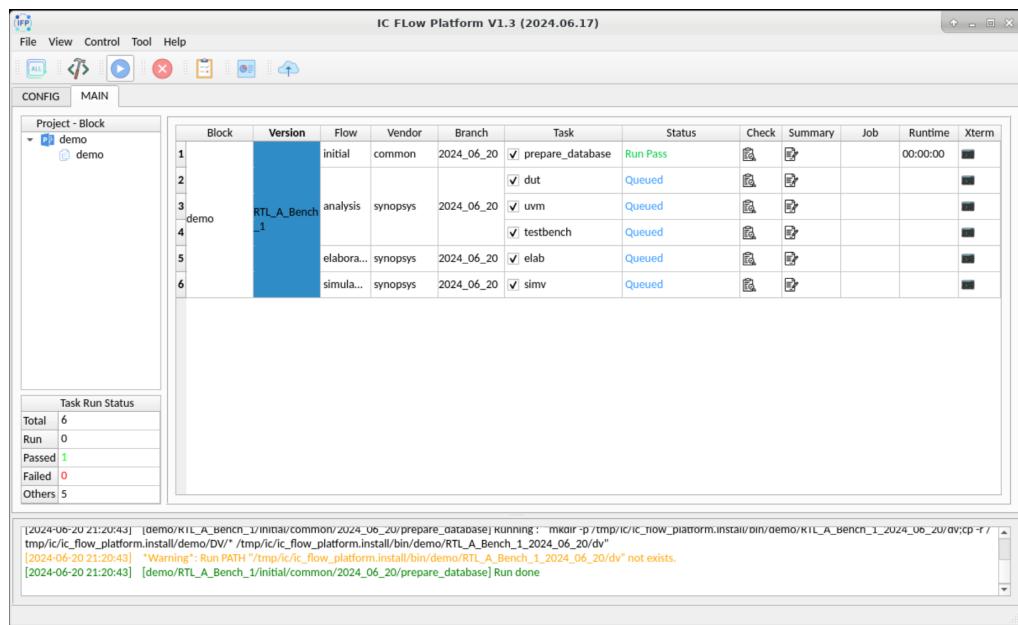


2.3 执行任务

(1) 在 *MAIN* 界面, *Version* 栏位 点击右键, 将弹出菜单, 选择 *Select All Task* 可选中该版本下的所有 Task:



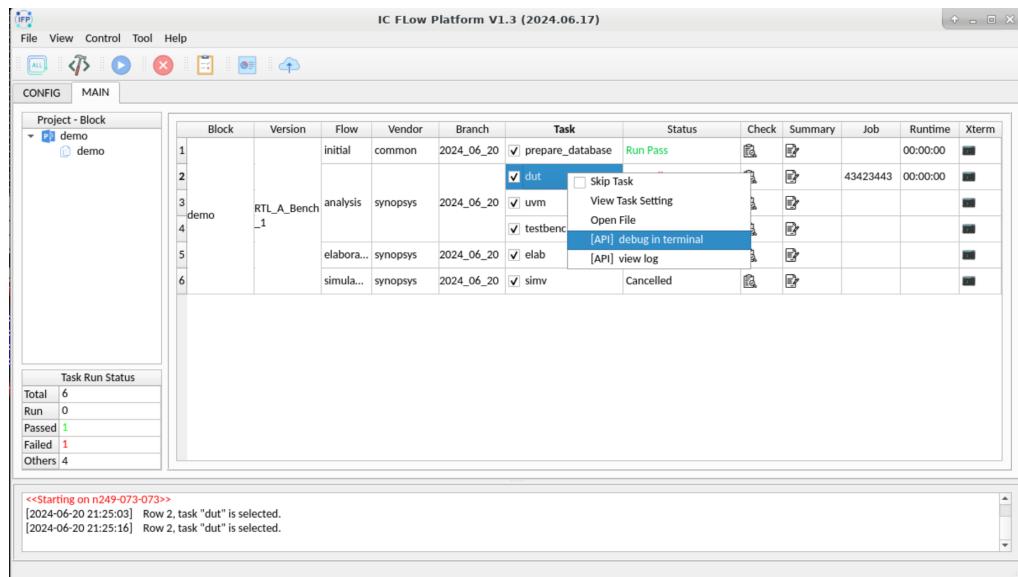
(2) 点击 **RUN** 按钮即可开始执行所有任务的 **RUN** 命令, 通过界面的 **Status** 栏位 观察 Task 状态:



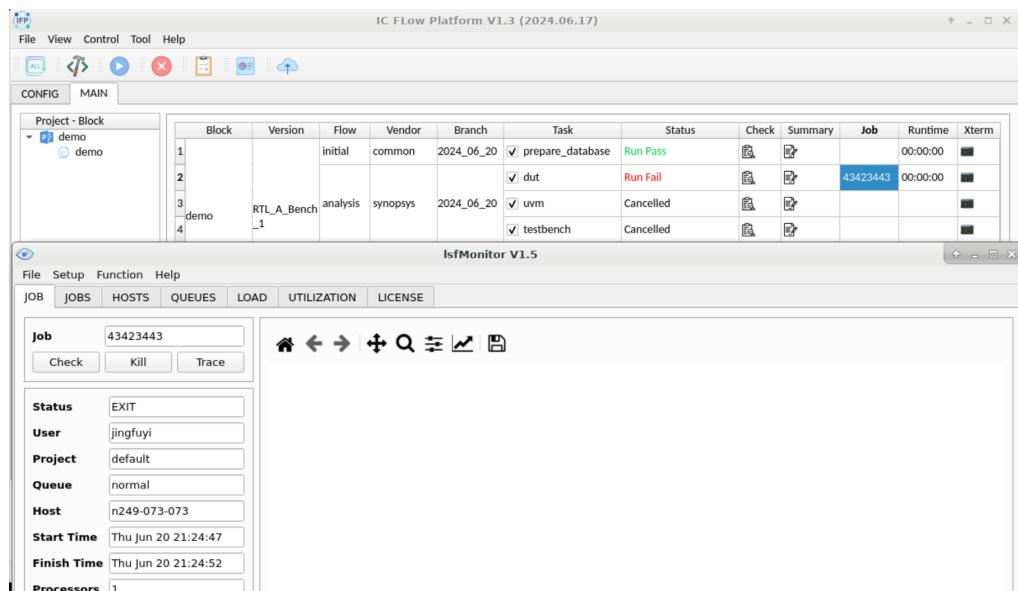
2.4 查看报告

可通过以下几种方式查看任务执行情况:

(1) 通过 API 打开 Terminal 并进入到指定路径或打开对应日志:



(2) 点击 Job ID 栏位, 可调用 IFP 安装包中的 *lsfMonitor* 软件进行 LSF 任务的分析:

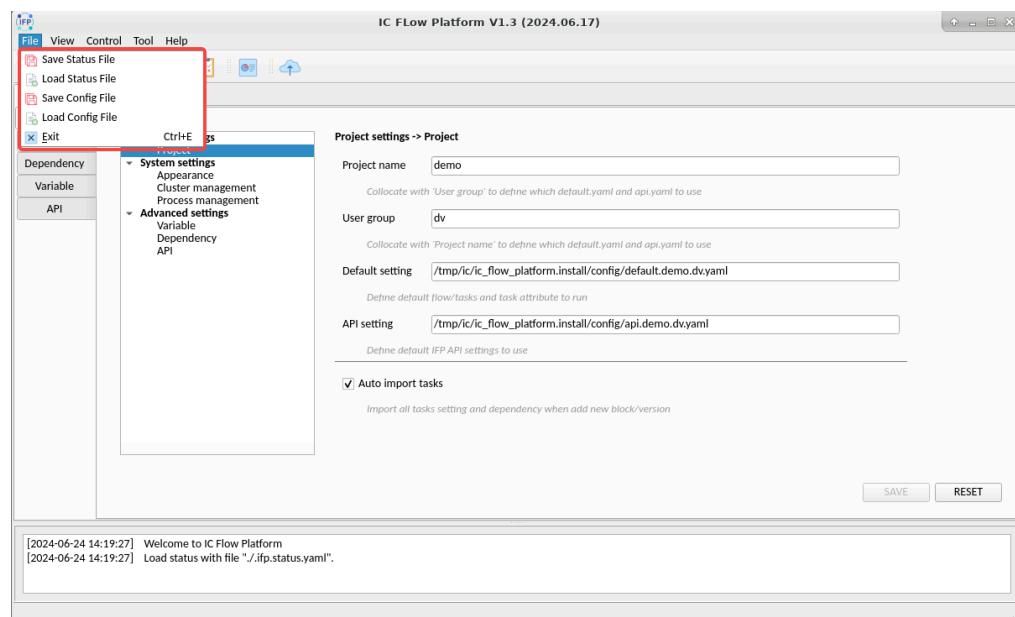


3. 功能介绍

3.1 菜单栏

菜单栏包括 *File/View/Control/Tool/Help*。

3.1.1 File



Save Status File

手动保存 *MAIN 界面* 中所有 Task 的状态信息到指定文件中。

Load Status File

手动加载指定文件中 Task 的状态到当前 *MAIN 界面*。若要选择隐藏文件，可在右键菜单中选中 *Show hidden files*。

Save Config File

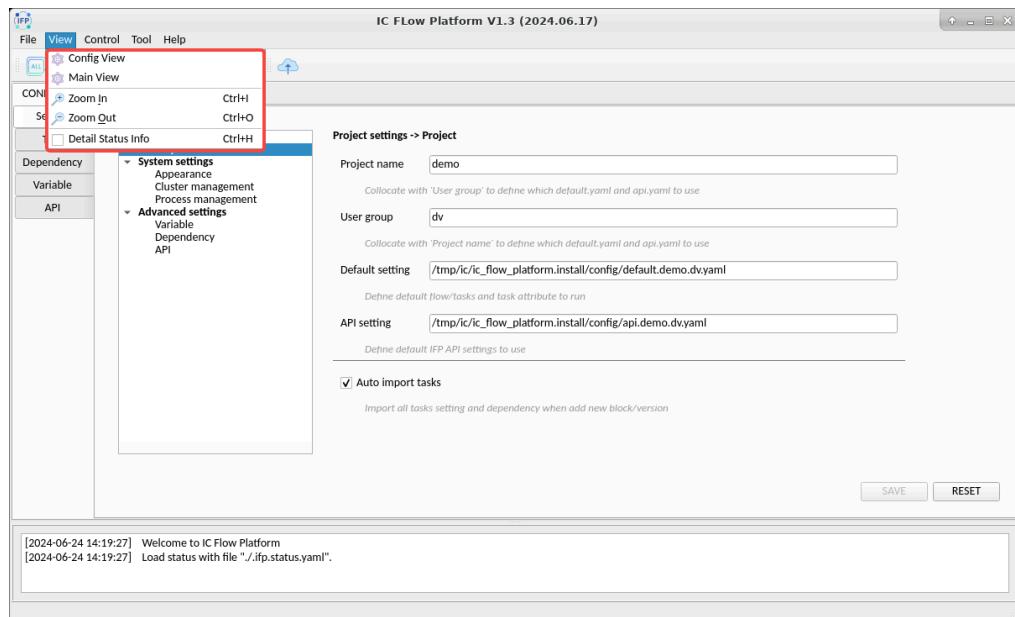
手动保存 *CONFIG 界面* 中的配置信息到指定文件中。

Load Config File

手动加载指定文件中的配置信息到 *CONFIG 界面* 中。若要选择隐藏文件，可在右键菜单中选中 *Show hidden files*。

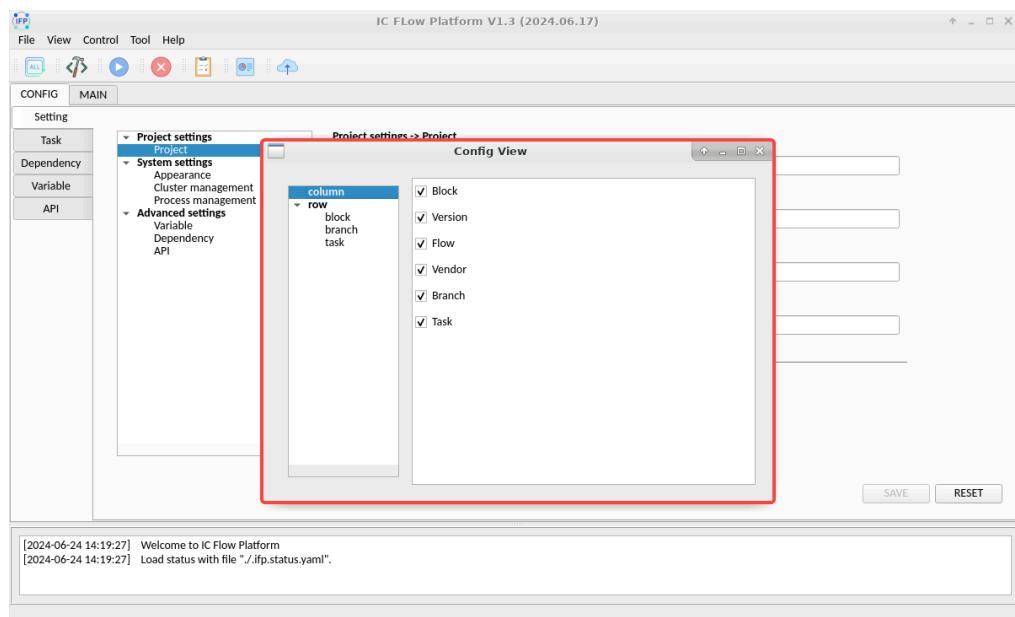
3.1.2 View

调整 IFP 的窗口大小和表格中显示的行列。



Config View

隐藏（显示）CONFIG 界面 指定的行或列。



Main View

隐藏（显示）MAIN 界面 指定的行或列

Zoom In

增加 IFP 窗口高度

Zoom Out

减小 IFP 窗口高度

Detail Status Info

MAIN 界面 可详细展示 Task 每一种 Action 的状态信息，快捷键为‘Ctrl + H’

The screenshot shows the IC Flow Platform V1.3 interface. On the left, there is a sidebar with tabs for CONFIG, MAIN, and Project - Block. Under Project - Block, there is a tree view with a single node 'demo'. The main area displays a table of tasks:

Block	Version	Flow	Vendor	Branch	Task	Build Status	Run Status	Check Status	Summarize Status	Release Status
1		initial	common	2024_06_24	<input checked="" type="checkbox"/> prepare_database	Pass				
2					<input checked="" type="checkbox"/> dut	Fail				
3	RTL_A_Bench_1	analysis	synopsys	2024_06_24	<input checked="" type="checkbox"/> uvm	Queued				
4					<input checked="" type="checkbox"/> testbench	Queued				
5			elaborate	synopsys	2024_06_24	<input checked="" type="checkbox"/> elab	Queued			
6		simulate	synopsys	2024_06_24	<input checked="" type="checkbox"/> simv	Queued				

Below the table is a summary table:

Task Run Status	Total
Total	6
Run	0
Passed	1
Failed	1
Others	4

3.1.3 Control

可进行不同 Action 的操作。

The screenshot shows the IC Flow Platform V1.3 interface with the Control tab selected. On the left, there is a sidebar with tabs for File, View, Control, Tool, and Help. Under Control, there is a dropdown menu with options: All_Steps, Build, Run, Kill, Check, Summarize, and Release. The 'All_Steps' option is highlighted with a red box.

The main area displays 'Project settings > Project' with the following fields:

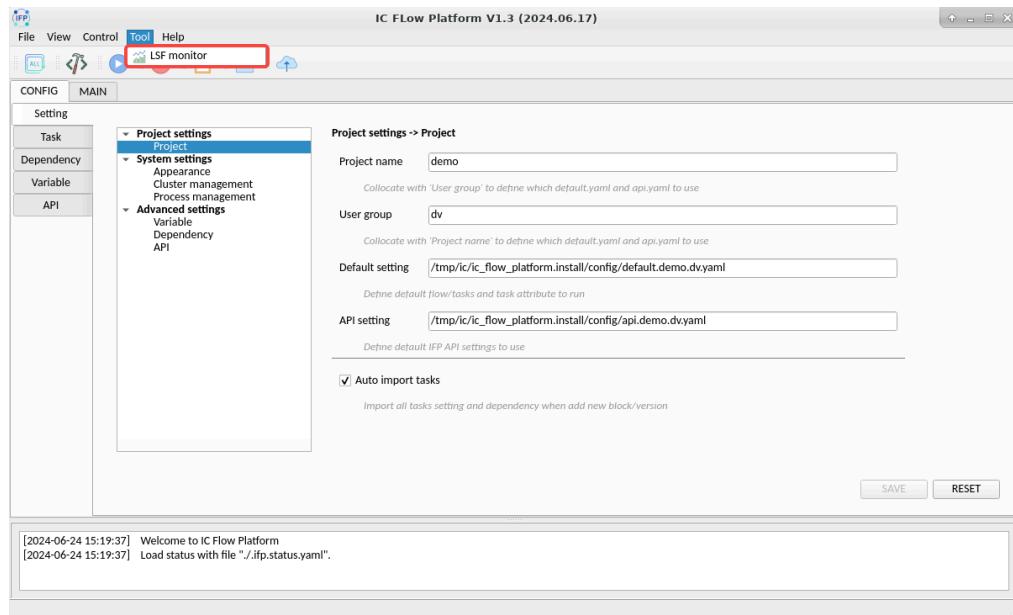
- Project name: demo
- User group: dv
- Default setting: /tmp/ic/ic_flow_platform.install/config/default.demo.dv.yaml
- API setting: /tmp/ic/ic_flow_platform.install/config/api.demo.dv.yaml
- Auto import tasks

At the bottom, there is a log window with the following text:

```
[2024-06-24 15:19:37] Welcome to IC Flow Platform
[2024-06-24 15:19:37] Load status with file "./ifp.status.yaml".
```

3.1.4 Tool

此处会不断集成 Bytedance 自研的开源工具来辅助用户利用 IFP 进行日常的工作。



LSF monitor

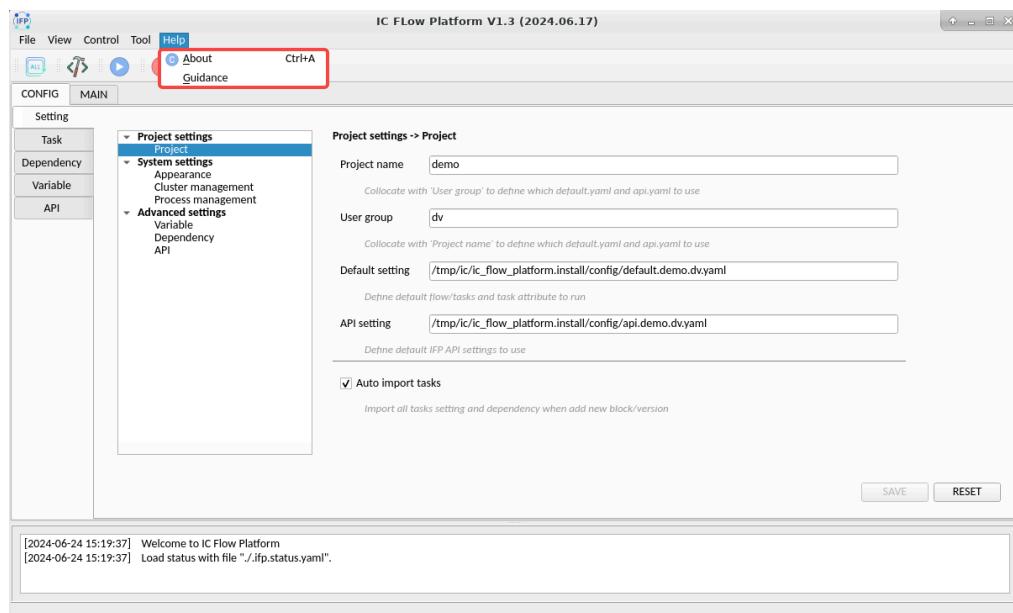
执行开源工具 *lsmMonitor*, 用于获取 LSF 的基本信息。

lsmMonitor 是一款用于 LSF/openlava 数据收集、分析及展示的开源工具, 亦可用于 EDA license 实时信息检索, 可以满足集成电路设计用户对于 LSF/license 的绝大部分信息查询和常规问题解决需求。

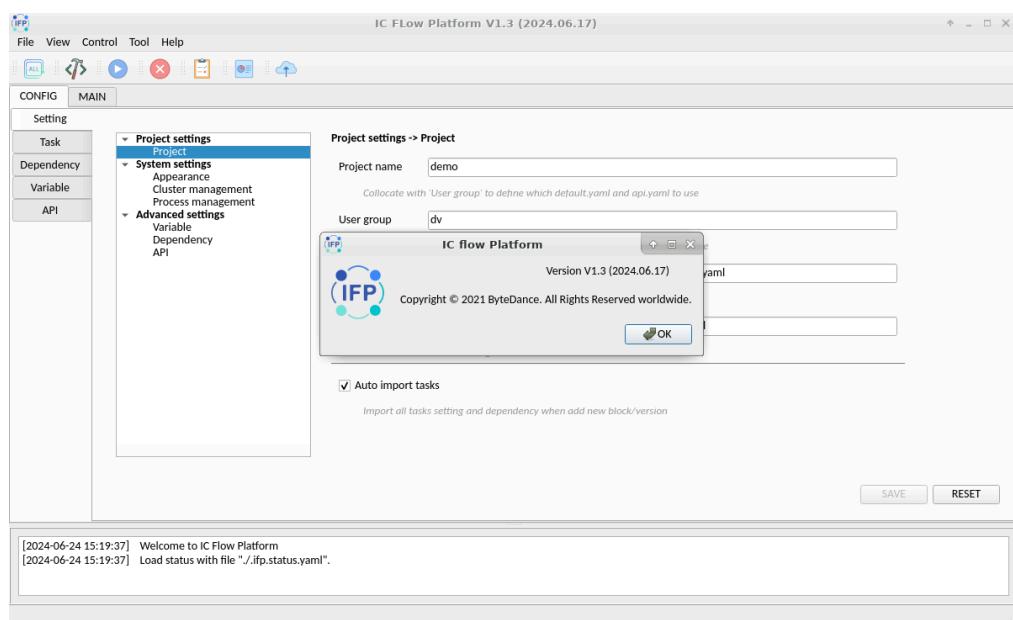
lsmMonitor												
File		Setup		Function		Help						
JOB		JOBS		HOSTS		QUEUES		LOAD		UTILIZATION		LICENSE
Job	User	Status	Queue	Host	Started	Project	Slot	Rusage (G)	Mem (G)	Command		
454 12267414		RUN	fpga	n212-204-144	2023-12-21 10:26:42	fpga	2	9.8	0.4	make -f /fpga/proj/fpga/...		
455 12268354		RUN	fpga	n212-204-144	2023-12-21 10:37:05	fpga	2	5	0.1	make sim >/dev/null 2		
456 9993565		RUN	orca	n232-132-209	2023-12-07 23:59:44	orca	8	78.1	2.7	gmake -f ./...		
457 10636187		RUN	orca	n232-132-209	2023-12-12 10:23:35	orca	4	100	1.2	pt_shell -f run.tcl		
458 11151929		RUN	orca	n232-132-209	2023-12-14 11:03:09	orca	8	50	13.8	/ic/software/synopsys/...		
459 11865191		RUN	orca	n232-132-209	2023-12-18 17:12:18	orca	8	43.6	19.7	fc_shell		
460 12189309		RUN	orca	n232-132-209	2023-12-20 18:25:07	orca	1		2.6	#!/bin/...		
461 12189310		RUN	orca	n232-132-209	2023-12-20 18:25:07	orca	1		2.7	#!/bin/...		
462 12195369		RUN	orca	n232-132-209	2023-12-20 19:33:34	orca	8	78.1	81.8	joules_studio -files ./scr...		
463 12266486		RUN	orca	n232-132-209	2023-12-21 10:07:51	orca	8	50	4.9	innovus -stylus -wait 12...		
464 10092228		RUN	vivian	n232-132-084	2023-12-08 10:27:40	vivian	1	2.2	1.1	./run -verdi		
465 10274869		RUN	vivian	n232-132-084	2023-12-09 15:58:47	vivian	1	7.2	5	verdi -ssf sim/sim/...		
466 11284317		RUN	vivian	n232-132-084	2023-12-15 10:29:24	vivian	1	4.9	3.1	verdi -ssf wave.vf		
467 11313489		RUN	vivian	n232-132-084	2023-12-15 14:31:45	vivian	1	1.1	1.5	wv		
468 11317944		RUN	vivian	n232-132-084	2023-12-15 15:15:00	vivian	1	0.6	1.5	verdi.run		

3.1.5 Help

帮助信息。

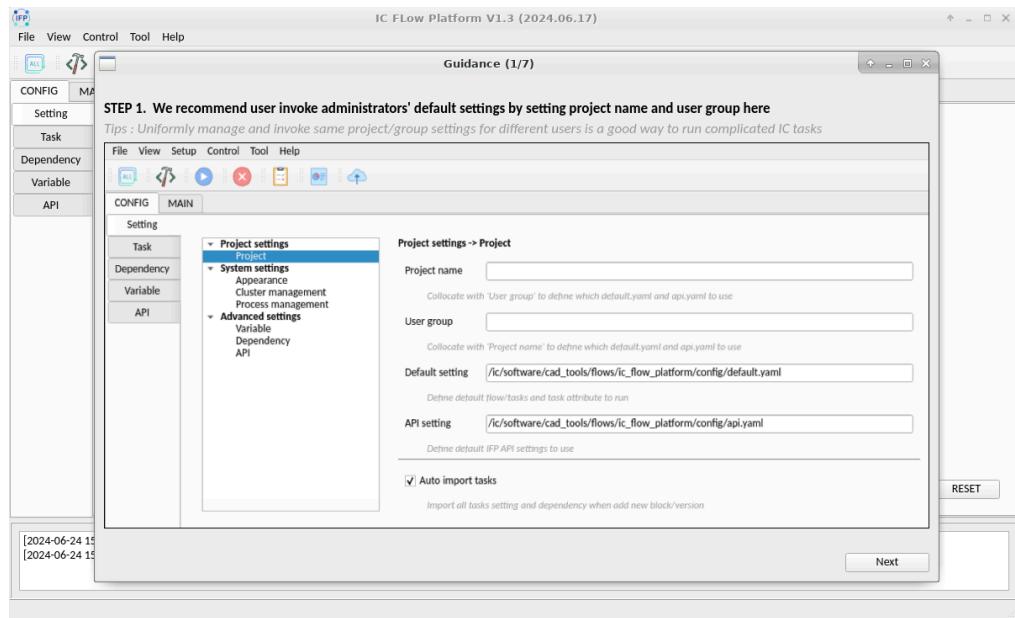


About



Guidance

查看 IFP 的主要功能介绍。



3.2 功能区

功能区包含了多个执行任务的按钮，用来控制 IFP 的运行。从左至右依次包括：

Run All Steps 依次运行 Task 的所有流程

Build 运行 Task 的 Build 流程，一般为创建该 Task 的运行环境（可选）

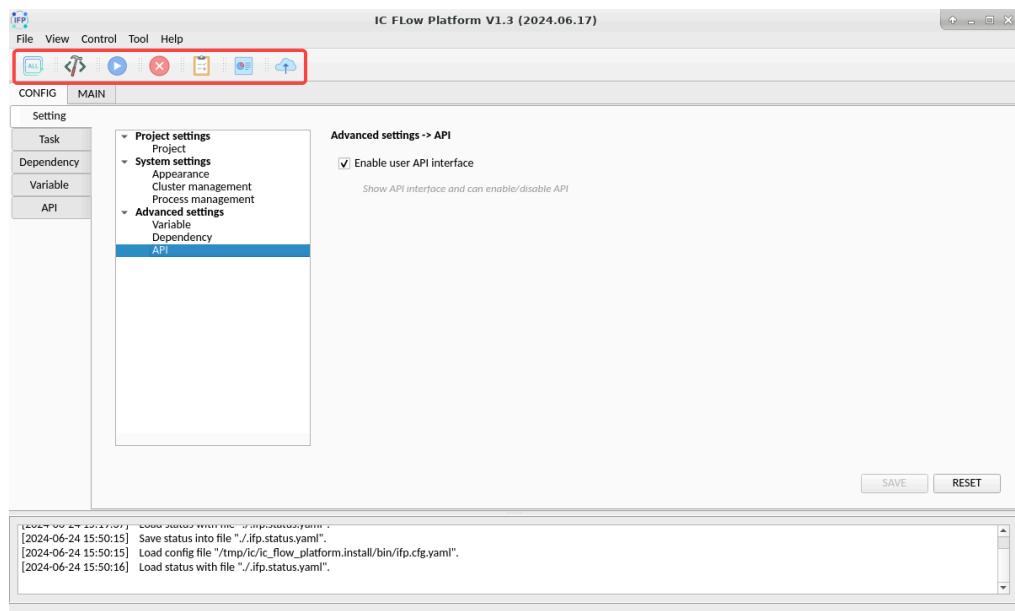
Run 运行 Task 的 Run 流程，一般为 EDA 的主体任务（必须设定）

Kill 终止当前的 Action

Check 运行 Task 的 Check 流程，例如检查 Run 的结果（可选）

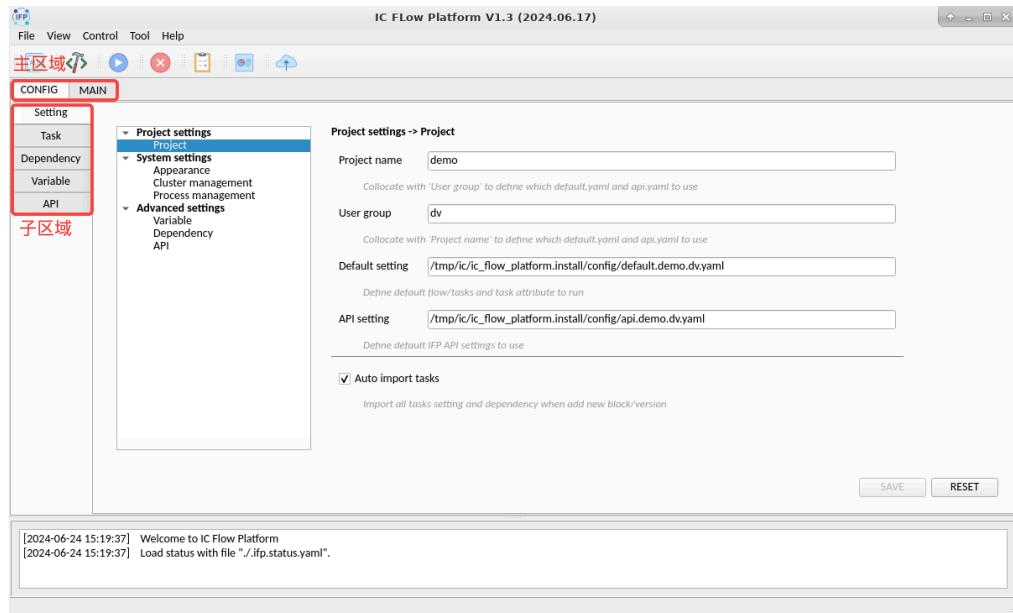
Summarize 运行 Task 的 Summarize 流程，例如总结报告（可选）

Release 运行 Task 的 Release 流程，例如发布数据（可选）



3.3 用户设置区域

CONFIG 界面 用于系统设置和任务设置。其中包含了 5 个子界面，分别是 系统设置 (*Setting*) / 任务设置 (*Task*) / 依赖关系设置 (*Dependency*) / IFP 内置变量设置 (*Variable*) / 应用程序编程接口设置 (*API*) 。

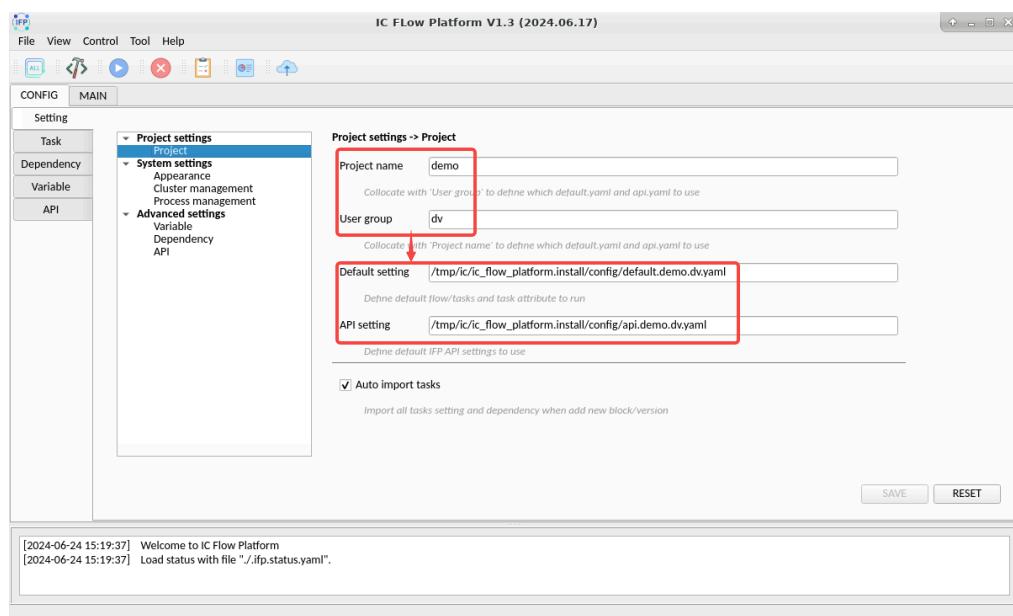


3.3.1 系统设置界面

控制 IFP 的外观、默认设置、执行逻辑等，包含 Project settings / System settings / Advanced settings 三部分。

Project settings

根据用户所在项目和用户组，匹配管理员的默认设置。



Project name：用户所在项目名称

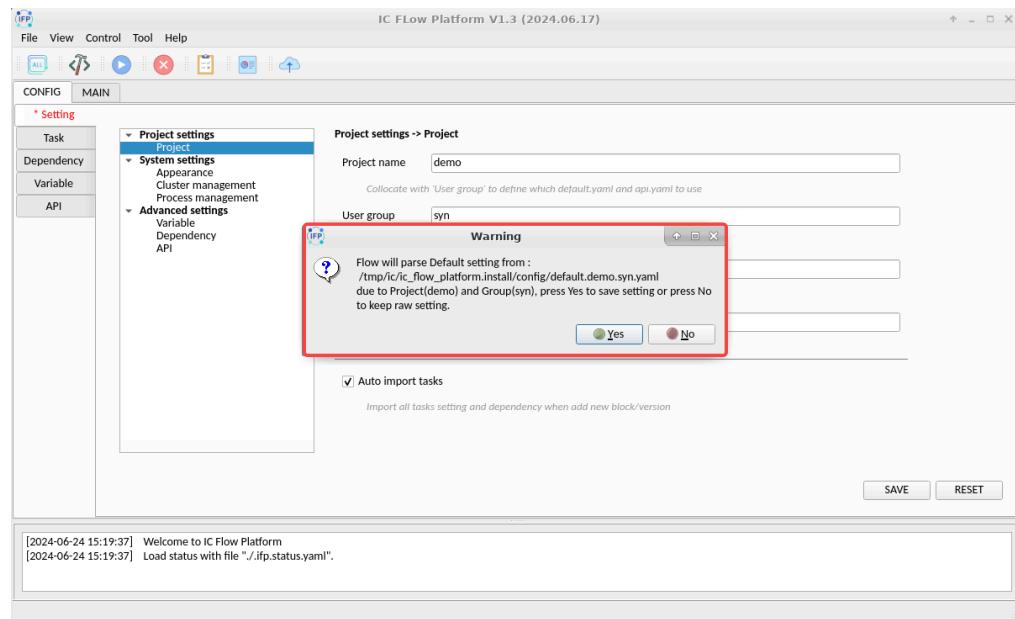
User group : 用户所在用户组

Default setting : 根据 *Project name* 和 *User group* 自动匹配管理员的默认流程设置

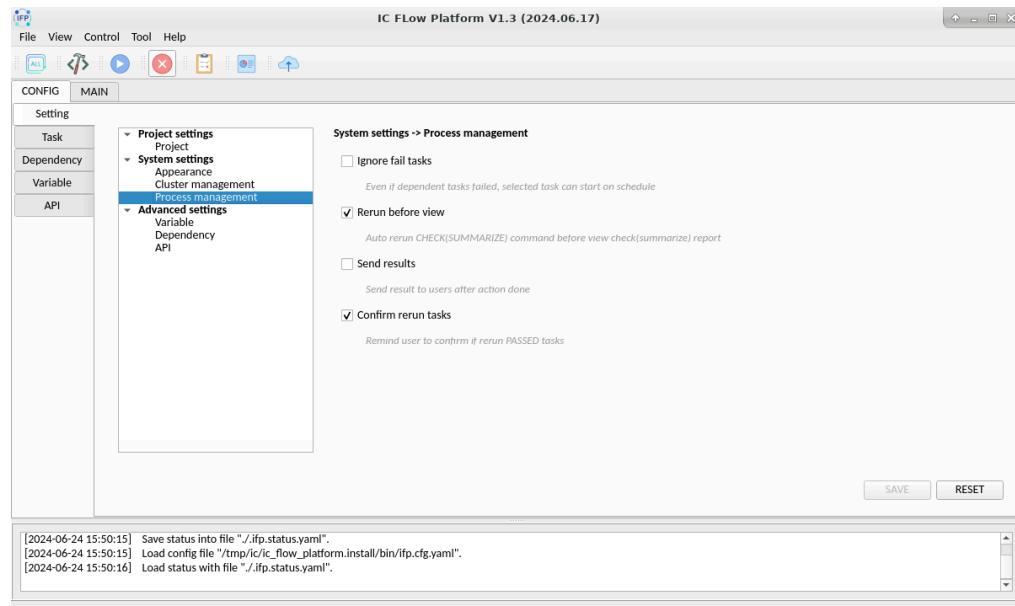
API setting : 根据 *Project name* 和 *User group* 自动匹配管理员的默认 API 设置

当用户更改 *Project name* 或 *User group* 时, 将按照如下的优先级匹配搜索默认设置, 并弹窗提示:

1. 按照路径优先级: \${HOME}/.ifp/config > \$<IFP_INSTALL_PATH>/config
2. 按照文件优先级: default.\${project}.\${group}.yaml > default. \${group}.yaml > default. \${project}.yaml > default.yaml



System settings

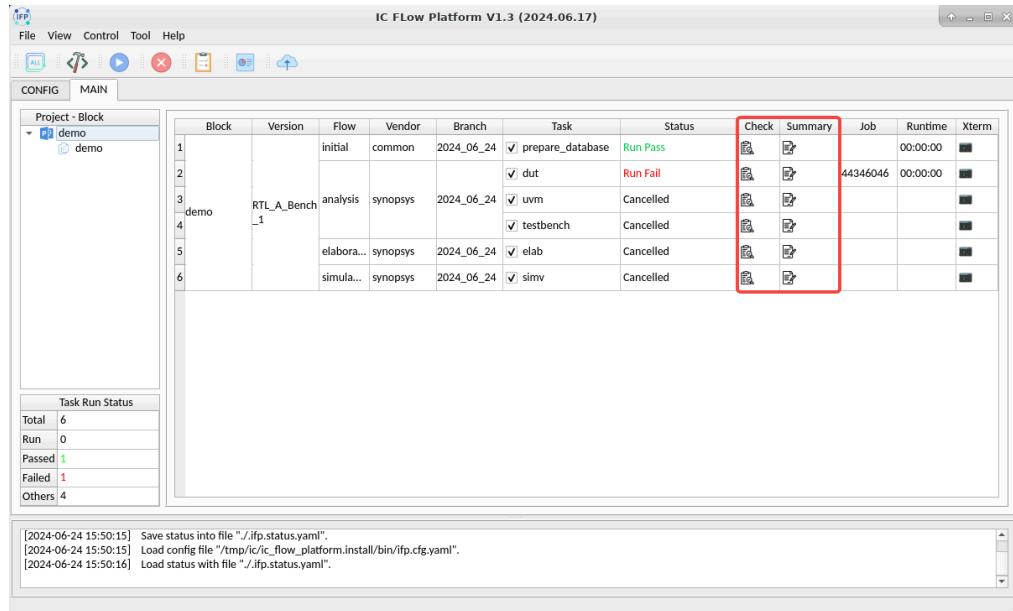


Fullscreen mode: 开启全屏模式

\$BSUB_QUEUE: 用 LSF 分发任务时调用的队列

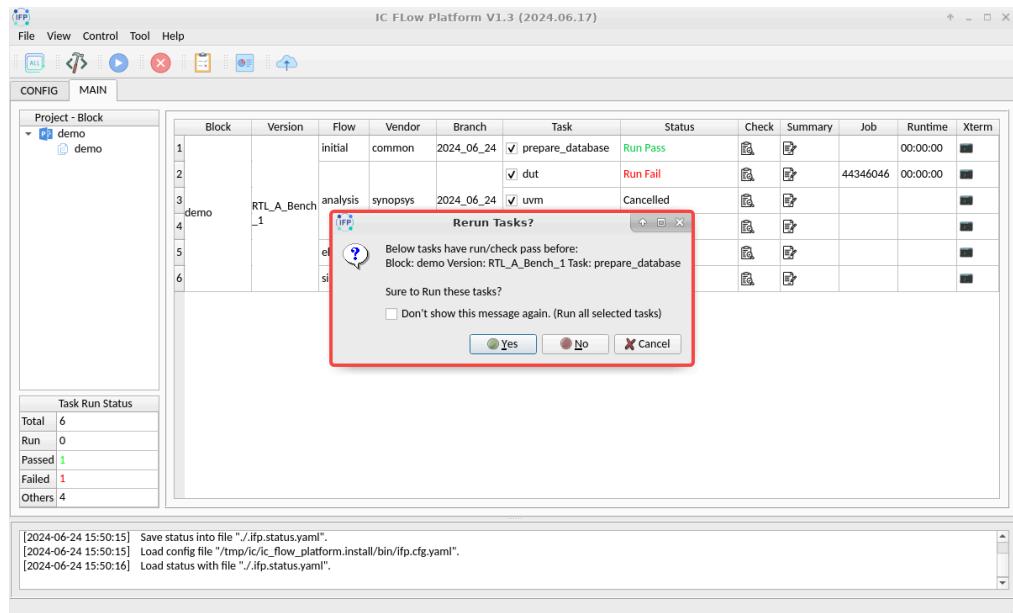
Ignore fail tasks: 若任务与任务之间存在依赖关系, 即使前置任务失败后, 后续任务继续提交

Rerun before view: 当用户点击 Check / Summary 查看报告时, 是否强制重跑一次 Check/Summarize Action



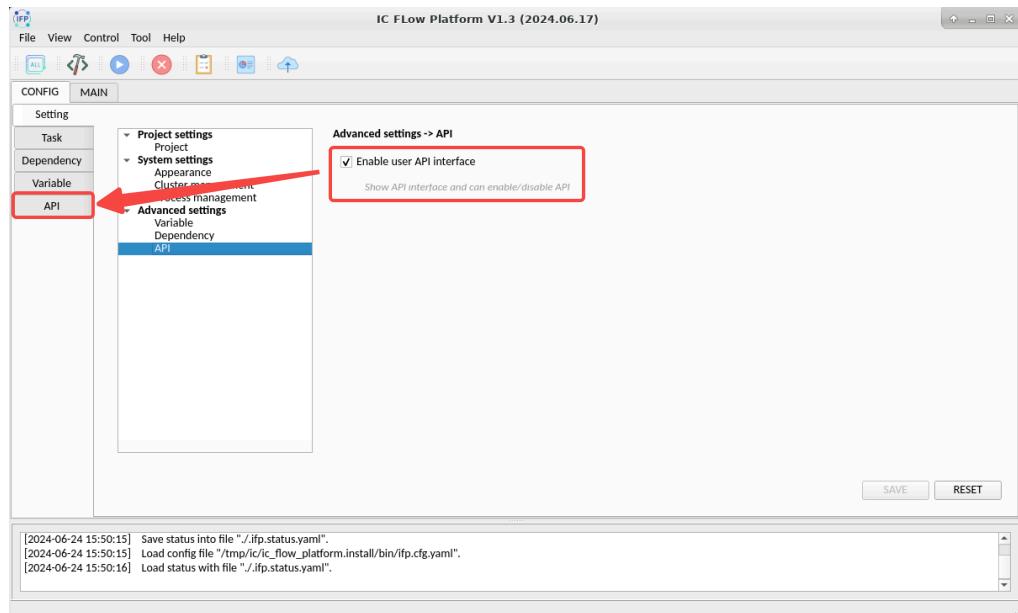
Send results: 任务结束后按照管理员指定的方式发送汇总结果给用户

Confirm rerun tasks: 对于状态为 Run Pass 的任务, 若用户再次点击 Run, 将会弹窗提示以确认是否需要 Rerun。若用户选择 Yes, 则将开始 Rerun。若用户选择 No, 则将跳过已经 Pass 的任务开始 Rerun 其他任务。若用户选择 Cancel, 则不会有任何操作。



Advanced settings

高阶设置默认关闭, 用户可在此启动相关子页面。



Enable user variable interface 启用 IFP 内置变量设置界面

Enable user dependency interface 启用依赖关系设置界面

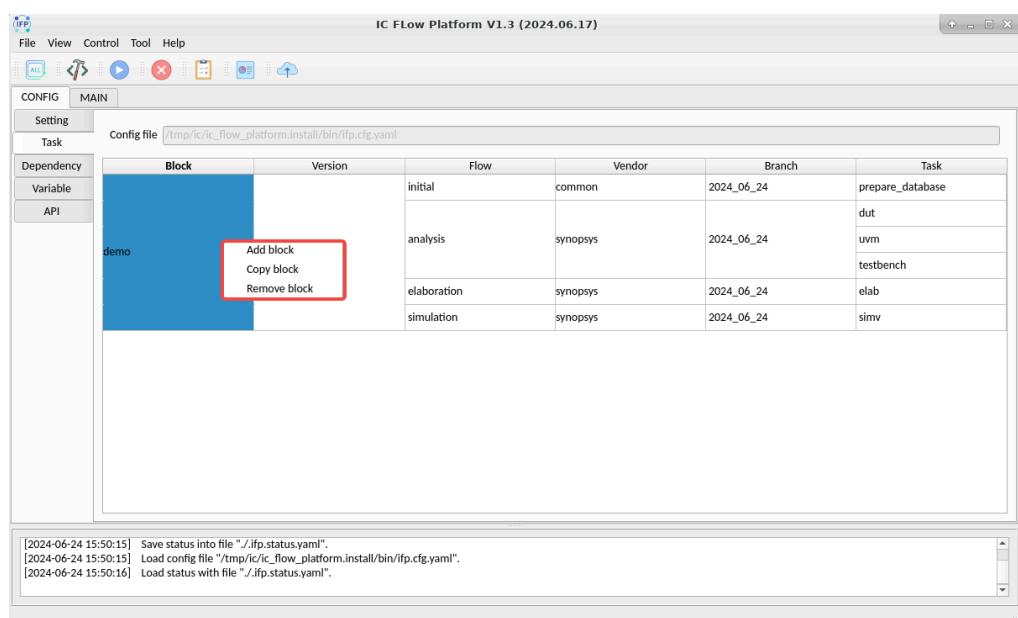
Enable user API interface 启用 API 设置界面

3.3.2 任务设置区域

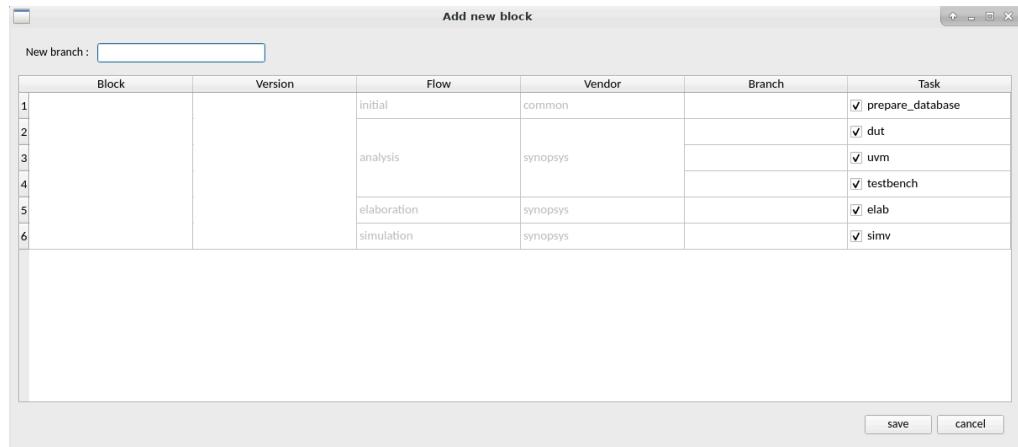
设置所需执行的任务以及调整任务属性

创建任务

在 Task 界面 右键菜单中包含了创建、复制、移除等命令，这些命令对不同列的对象会有所区别。



create block : 创建 block (需在表格空白处右击)。IFP 会自动导入管理员配置的默认流程设置，用户仅需在空白处双击填写 Block/Version/Branch 信息，并选择需要导入的 Task 即可。另外可以在 *New branch* 处批量定义 Branch 的名字。



Add block : 新增 block

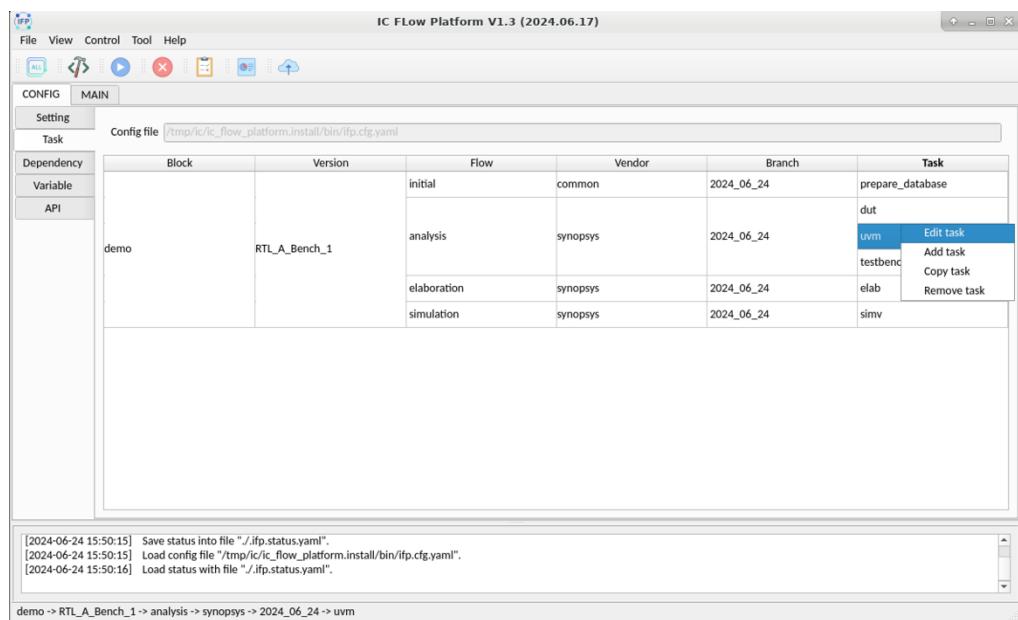
Copy block : 复制当前 block, 但需提供一个新的 block 名字

Remove block : 删除当前 block

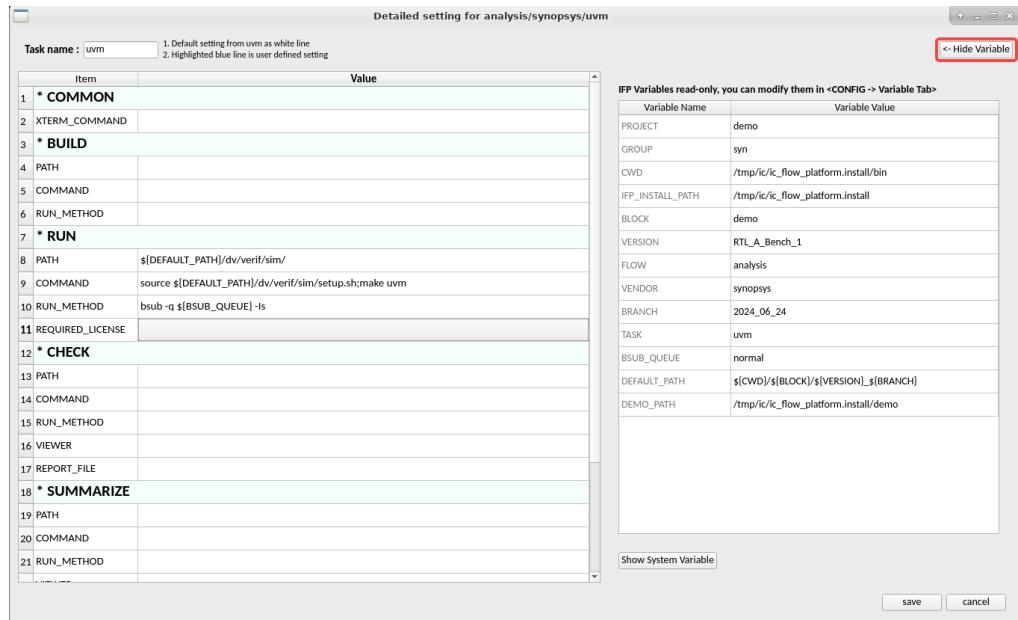
Version/Flow/Vendor/Branch/Task 列 具有类似的功能, 此处不再赘述。

调整任务属性

通过 Task 右键菜单中的 *Edit Task* 可查看和调整每一个具体 Task 的命令和参数。



在 Task name 的位置可以修改当前 Task 的名字, 但同时也会导致匹配到的管理员默认设置发生变更。点击 *Show Variable* (*Hide Variable*) 可显示 (隐藏) 相关的 IFP 内置变量。



左侧任务属性将按照如下的优先级进行选择：用户自定义配置 > 管理员默认配置 > 空。

当用户调整了属性时，将显示为蓝底黑字，此时若用户希望恢复默认，可删除自定义内容；

7	* RUN	
8	PATH	`\${DEFAULT_PATH}`
9	COMMAND	make syn_dc
10	RUN_METHOD	bsub -q \${BSUB_QUEUE} -ls -n 4

当用户未自定义属性并且管理员定义了默认属性时，将显示为白底黑字；

* RUN		
PATH		`\${DEFAULT_PATH}`
COMMAND		make syn_dc
RUN_METHOD		bsub -q \${BSUB_QUEUE} -ls

当用户与管理员均未定义属性时，将显示为空白（如 RUN_MOTHOD 处）。

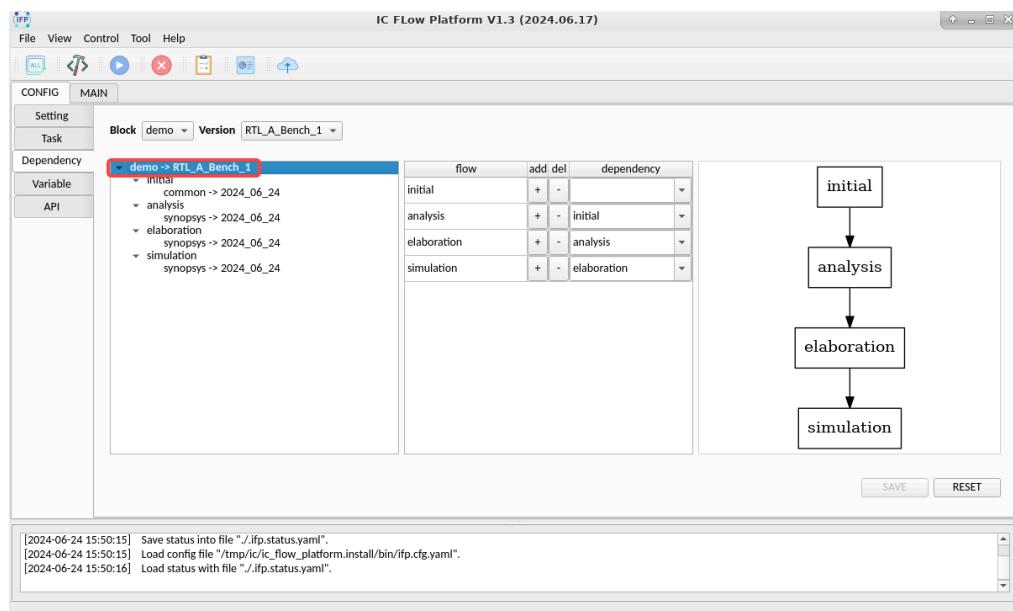
7	* RUN	
8	PATH	`\${DEFAULT_PATH}`
9	COMMAND	make syn_dc
10	RUN_METHOD	

3.3.3 依赖关系设置区域

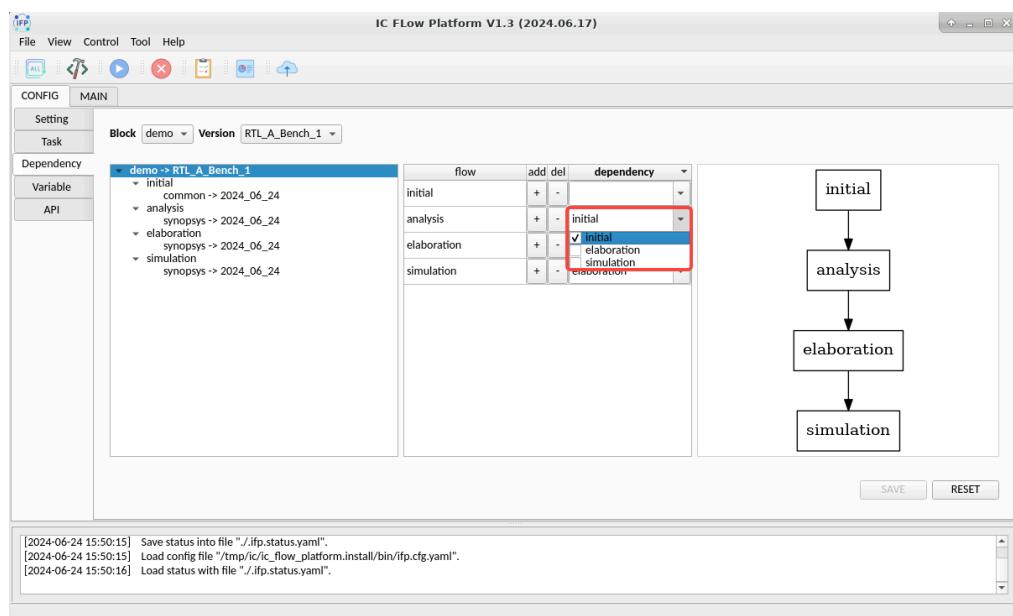
查看和调整管理员配置的默认依赖关系。IFP 支持串行、并行和多次触发的依赖关系。

设置 Flow 之间的依赖关系

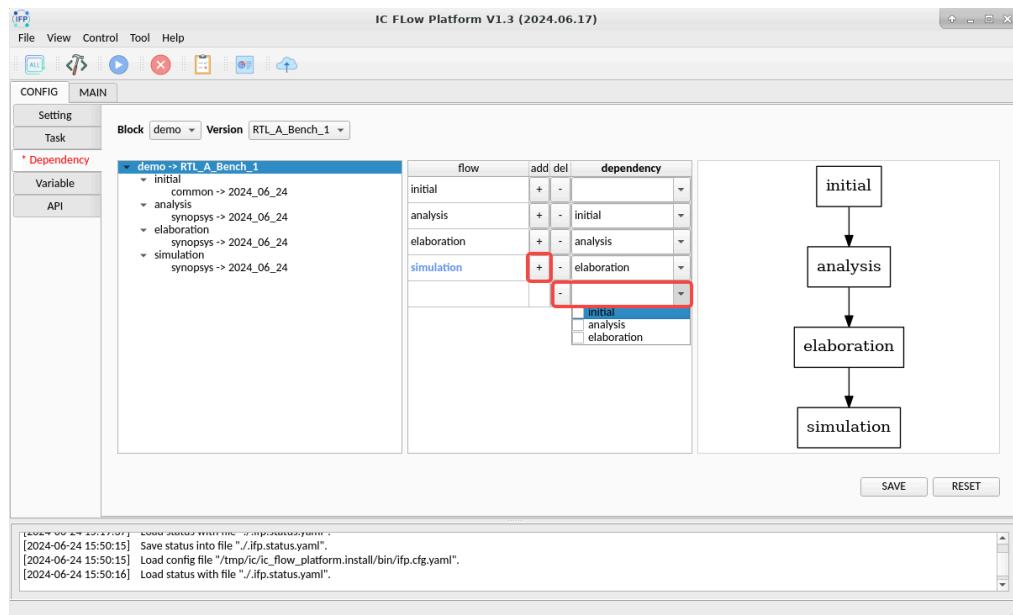
点击 *Block-Version* 层级即可开始设置 Flow 之间的依赖关系，同时右侧将实时展示 Flow 之间关系的树状图。



在 `dependency` 栏位 打开下拉框可选择该任务需在多个前置任务都完成后才可启动。

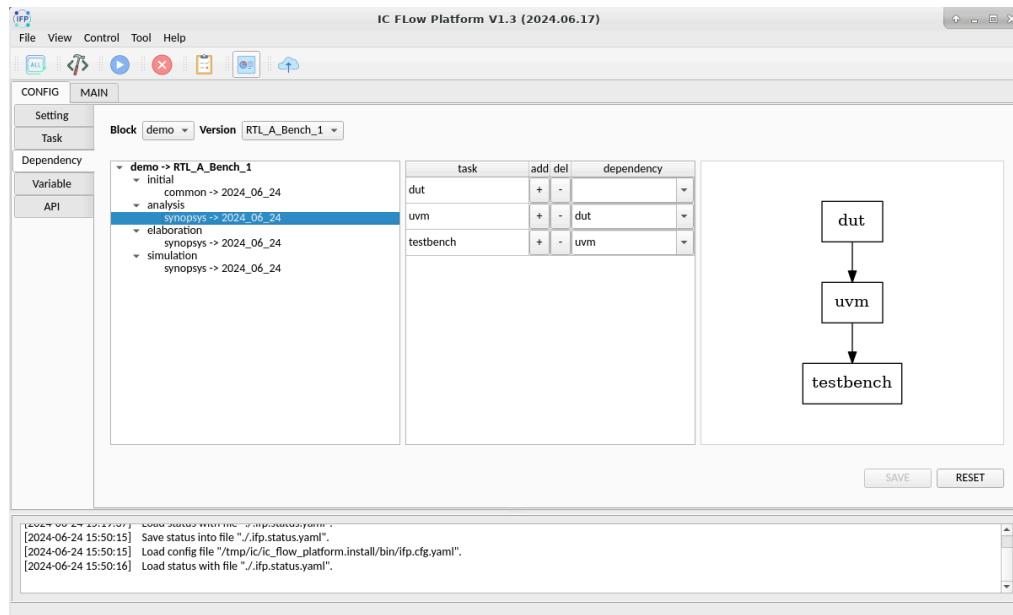


通过 `add (del)` 栏位 可添加 (删除) 多种场景，只要其中一个场景完成即可启动当前任务。



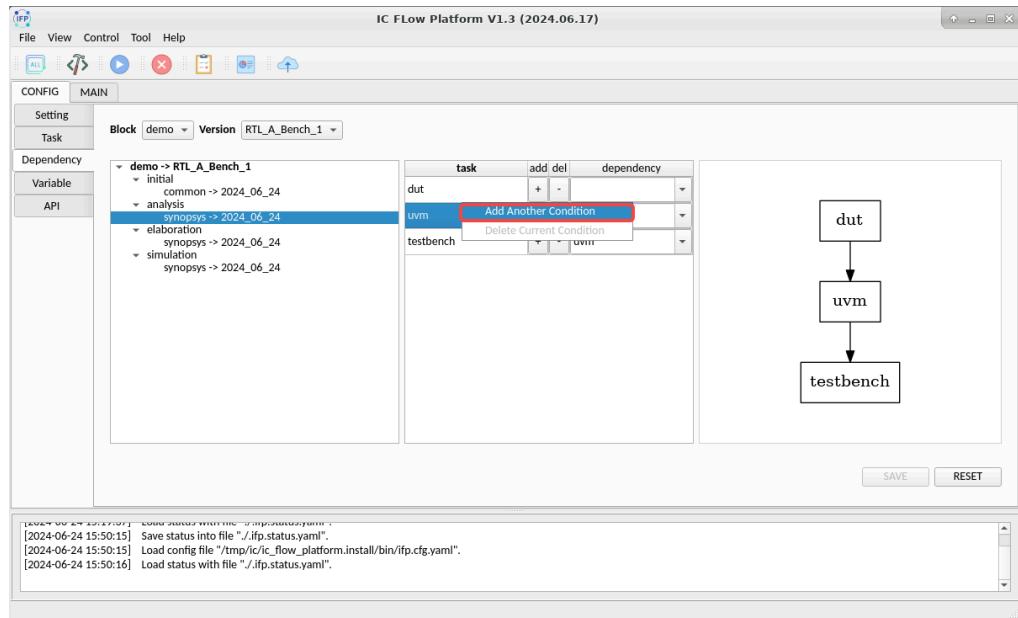
设置 Task 之间的依赖关系

点击 *Vendor/Branch* 层级即可开始设置 Task 之间的依赖关系，同时右侧将实时展示 Task 之间关系的树状图。



除了上述功能以外，Task 还存在一种较为特殊的依赖关系，即多次触发。

通过在 task 处右键 *Add Another Condition* 菜单可创建多种场景，每一种场景完成后都可启动一次当前任务。

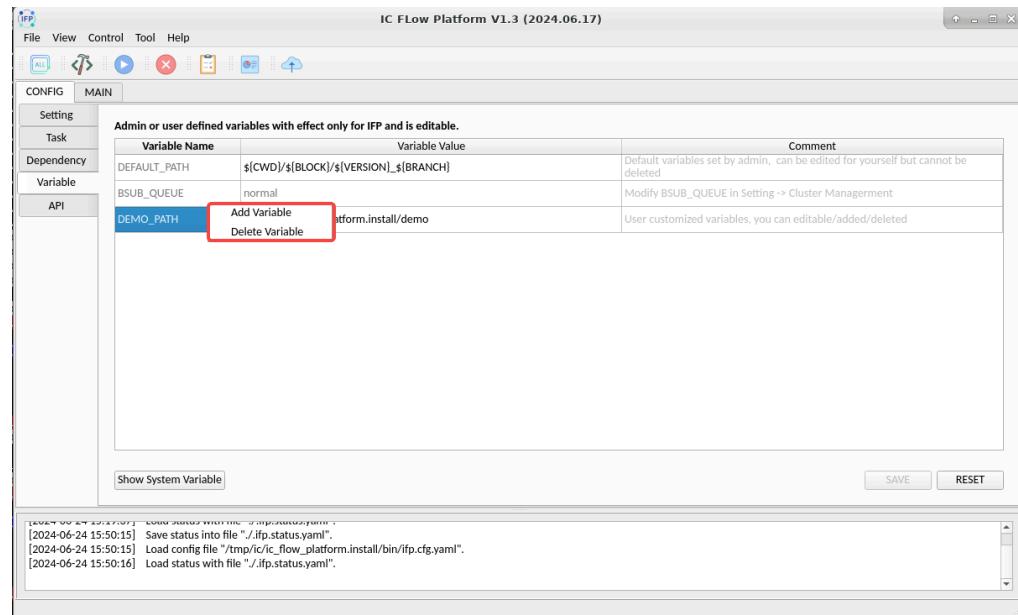


3.3.4 内置变量设置区域

查看和调整 IFP 的内置环境变量和系统环境变量，这些变量可在配置 Task 命令时直接调用（详见[章节 3.3.2 调整任务属性](#)）。其中变量包含多种类型：

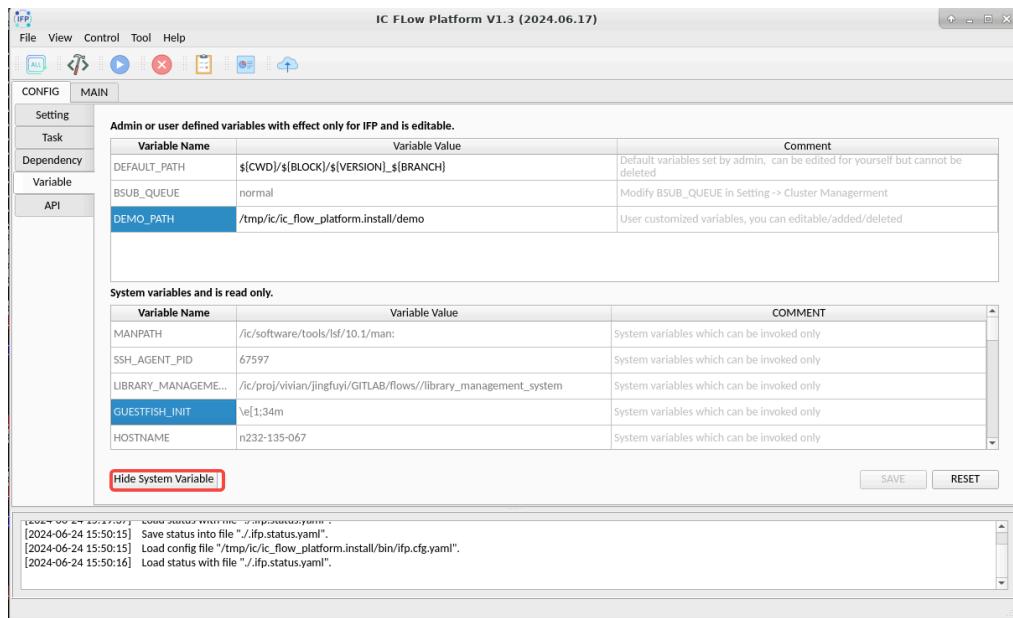
可编辑变量

管理员配置的默认环境变量、*Setting* 界面中的\$BSUB_QUEUE、用户自定义的 IFP 内置变量



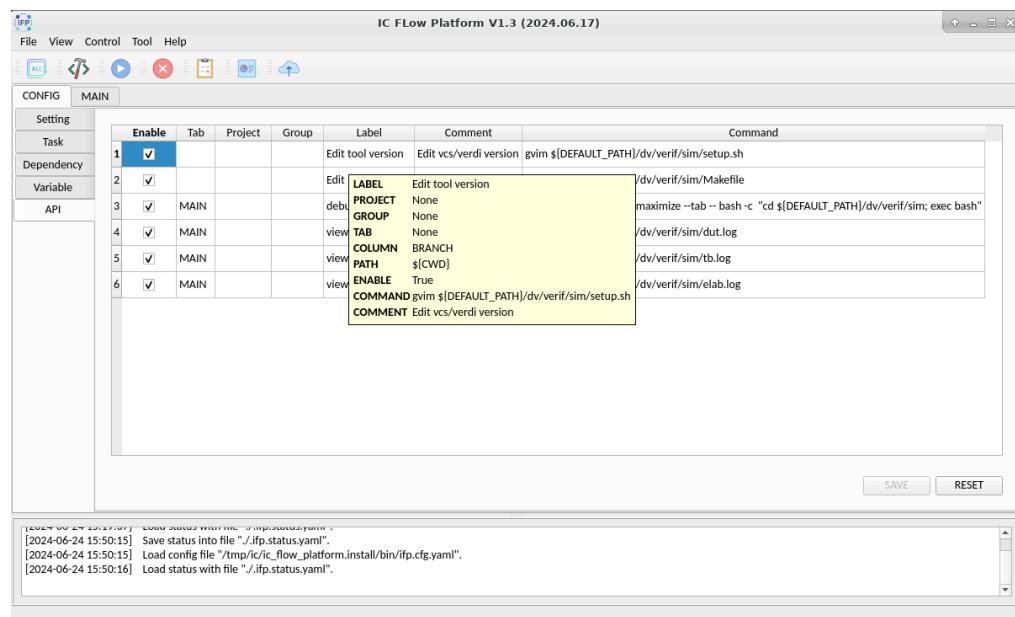
不可编辑变量

操作系统环境变量不可在 IFP 内部变更，请在启动 IFP 前或 \${IFP_install_path}/bin/ifp 中设置。



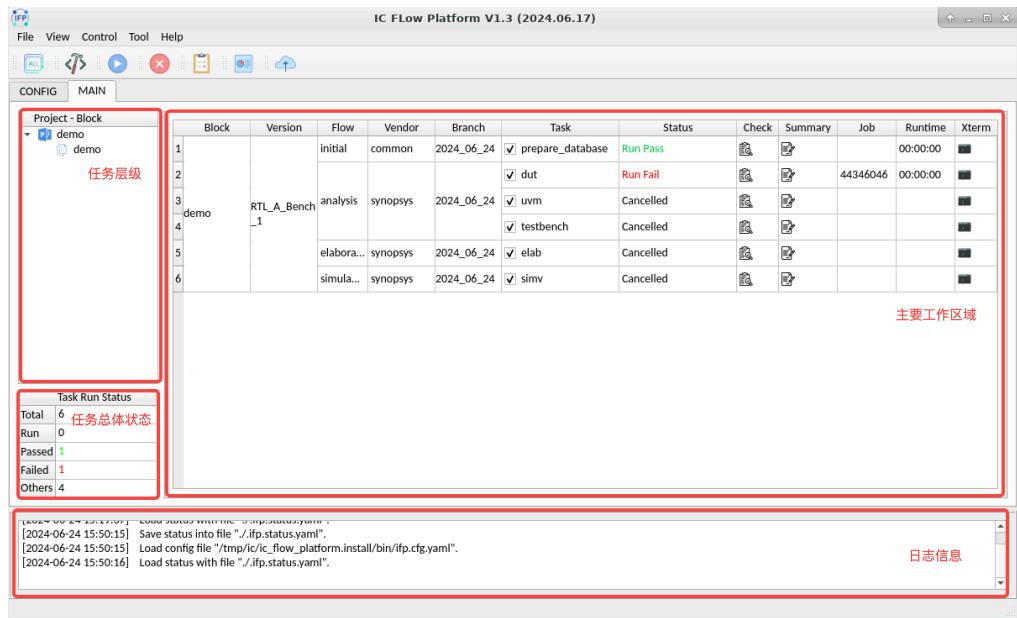
3.3.5 应用程序接口设置区域

查看和启用管理员配置的 API 功能。鼠标移动至 *Label* 栏位时将浮窗展示详细信息，包括该 API 生效范围、具体命令等。



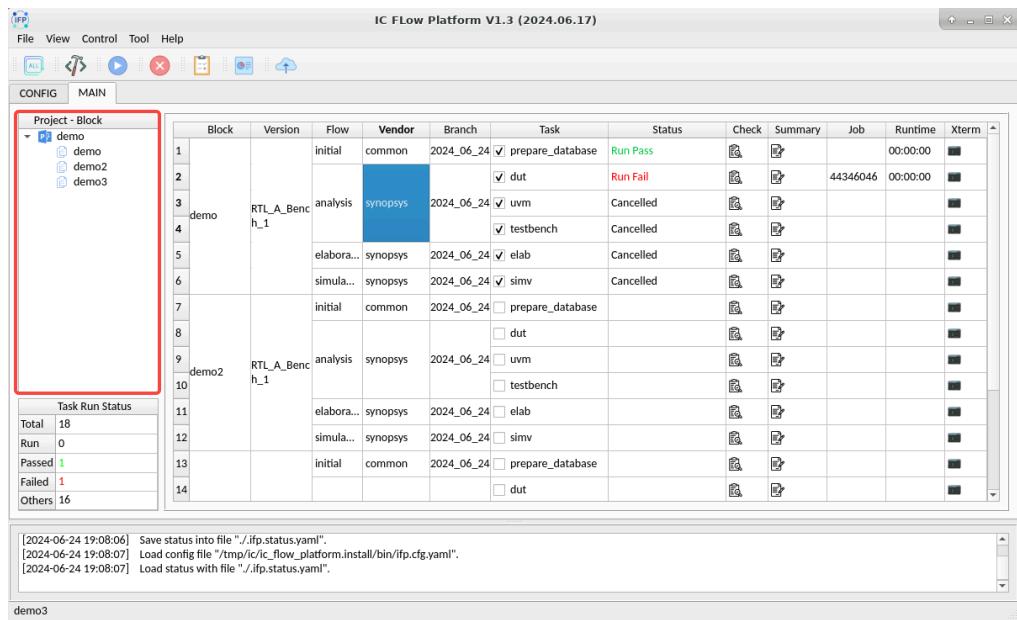
3.4 任务监控区域

MAIN 区域用于执行和监控相关的任务，搭配管理员设置的 API，可在同一个界面上集成日常工作所需的辅助工具，从而实现一站式的工作平台。



3.4.1 任务层级区域

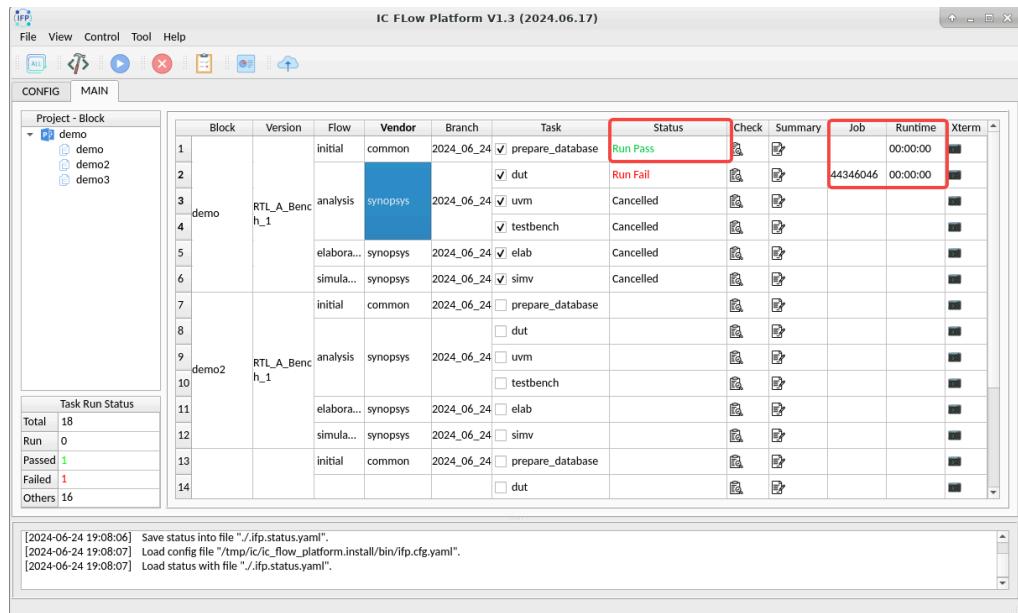
用于展示当前项目所需要处理的所有 Block。



3.4.2 主要区域

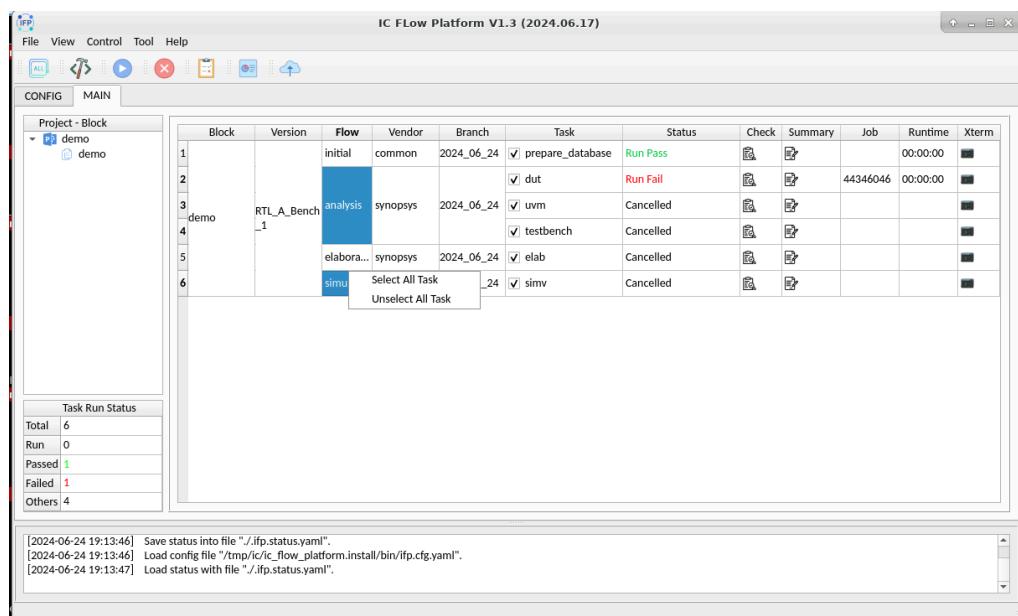
展示信息

除了具体任务信息以外，还会提供任务状态、任务 ID、执行时间等

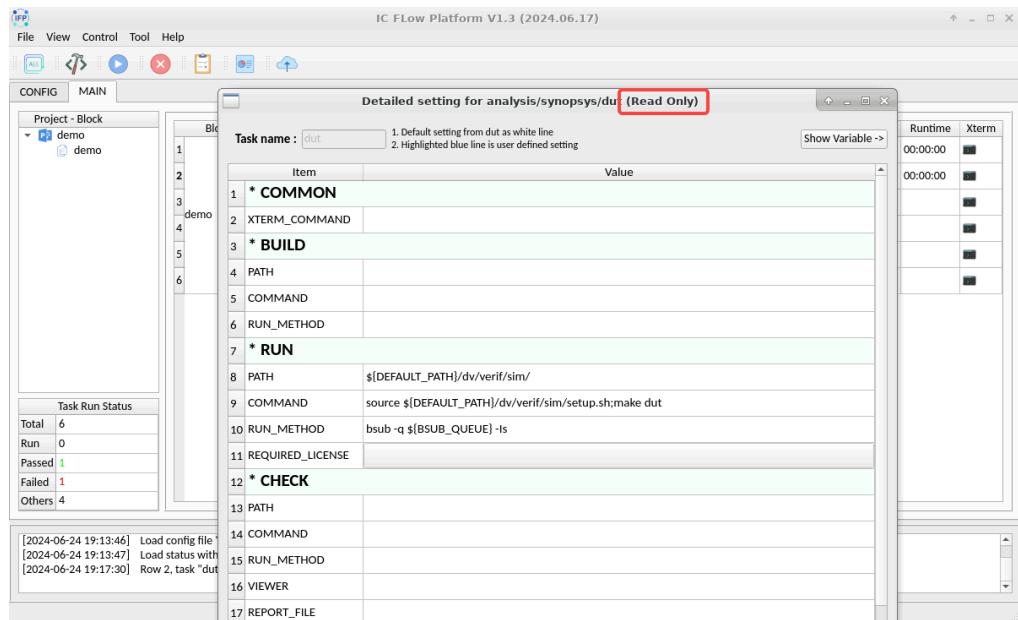


菜单功能

选择单个或多个 Block/Version/Flow/Vendor/Branch 时，可通过右键菜单中的 *Select All Task (Unselect All Task)* 来选中（取消）对应的 Tasks。



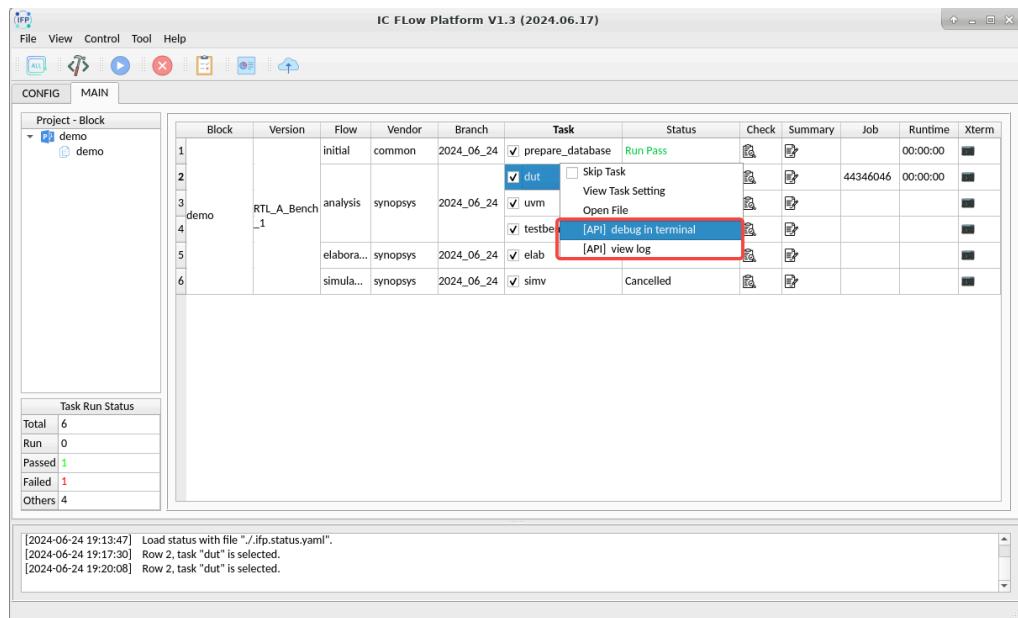
Task 右键菜单中可查看 Task 的详细属性（详见 [章节 3.3.2 调整任务属性](#)），但为只读模式。用户如需调整，可到 CONFIG 界面的 Task 界面 中进行编辑。



用户可通过勾选 *Skip Task* 来跳过某些暂时不想启动的 Task，但仍保持所有选中的 Task 之间的依赖关系链。

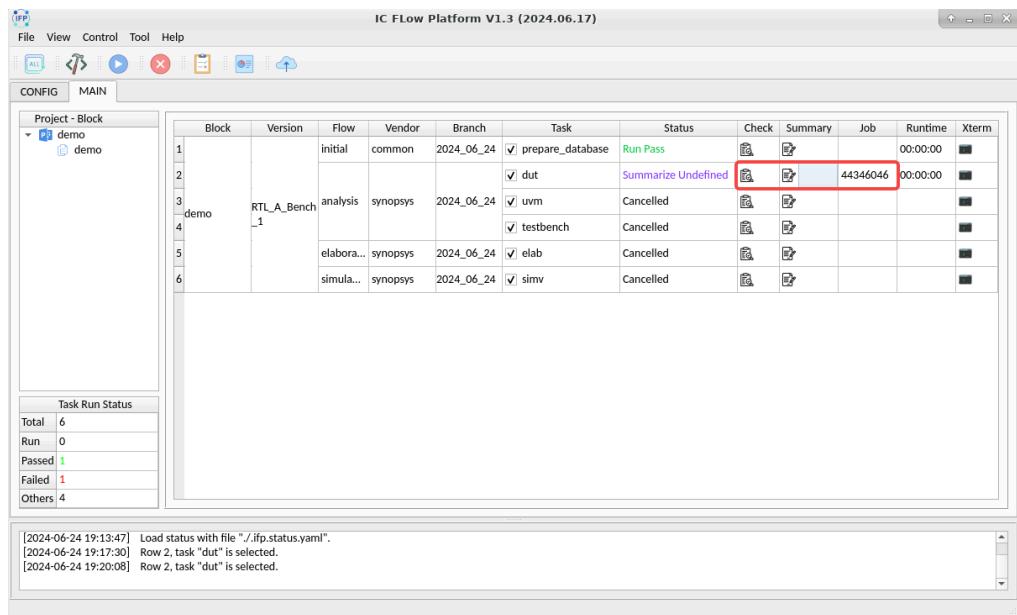
请注意，若用户取消勾选某个 Task，将会导致相关的 Task 关系链断裂！

若管理员设置了右键菜单的 API 功能，可在此调用相关功能，简化日常工作。例如打开终端，并自动执行某些命令，从而快速 Debug；例如快速打开对应 Task 的日志。

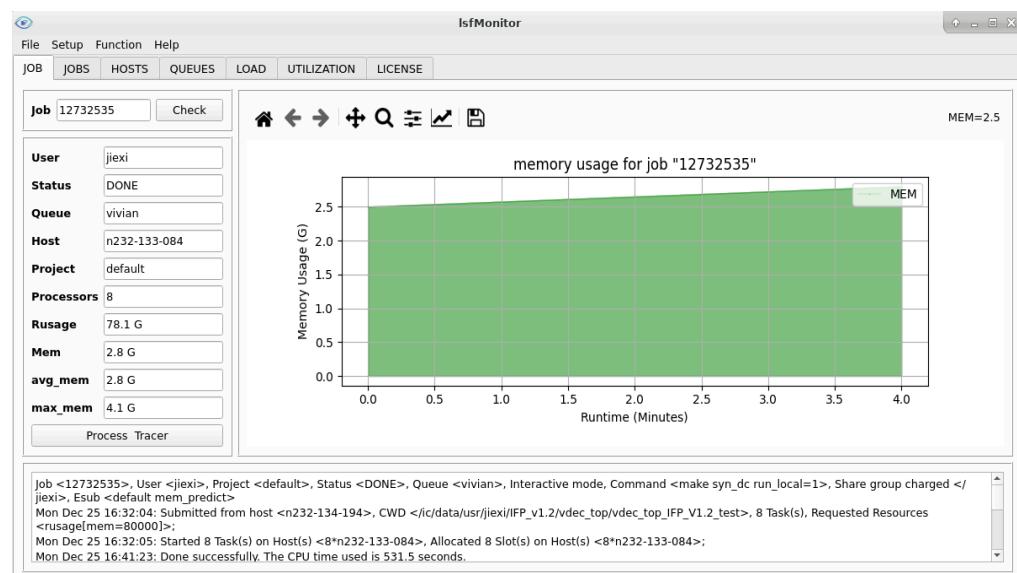


按钮功能

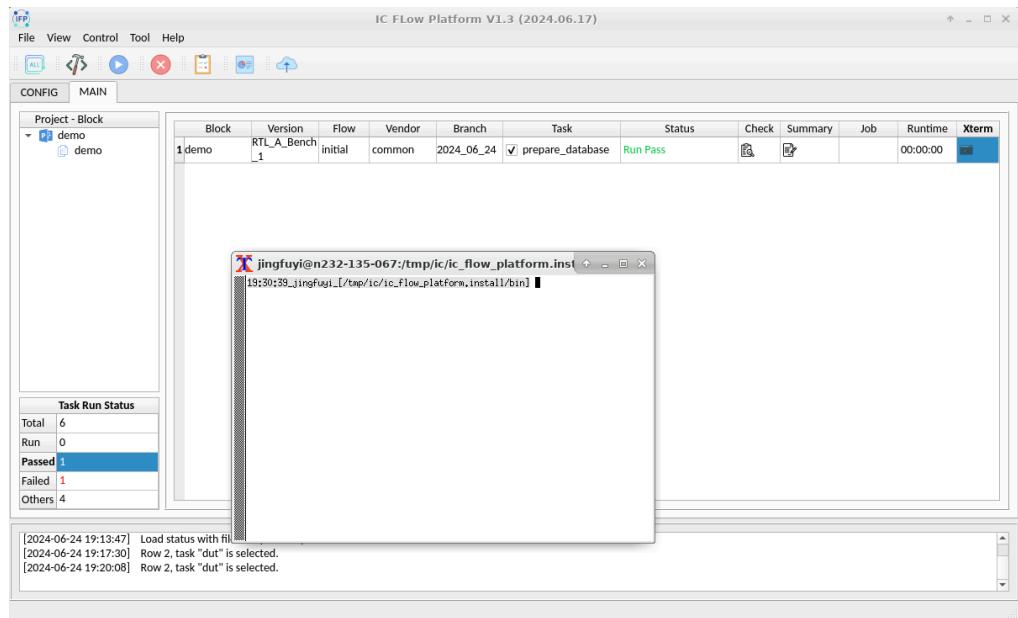
点击 *Check* 图标 将打开对应的 check 报告，若用户定义了 *Rerun before view*，则会先执行 Check 命令后再打开报告。
Summarize 图标 功能类似。



点击 *Job ID* 栏位 将调用 IFP 内置工具 *lsmMonitor* 来展示该 JOB 的详细信息。

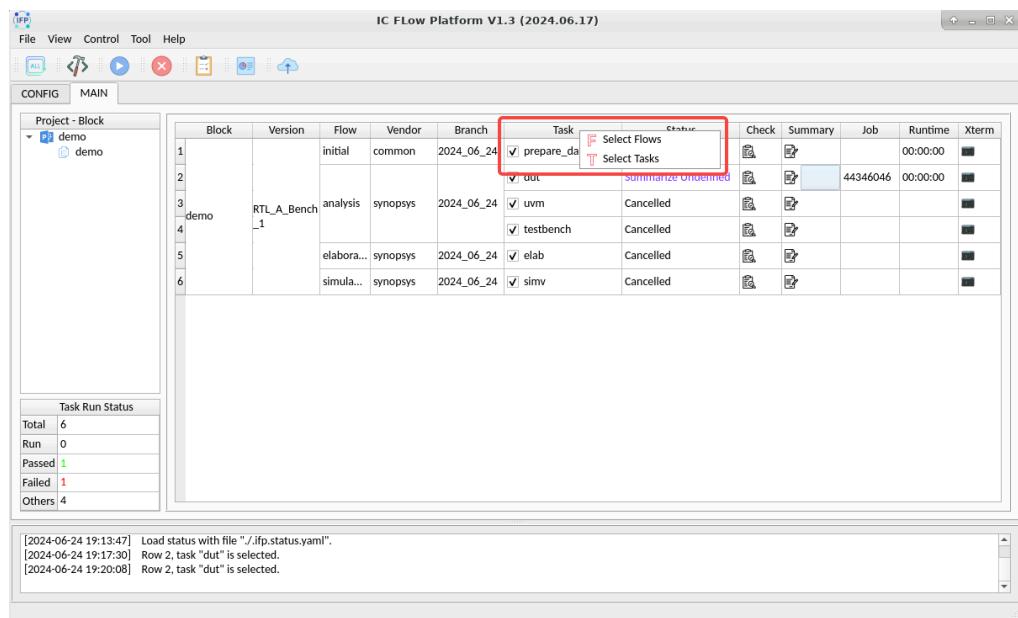


点击 *Xterm* 图标，将弹出一个终端，并自动进入该 Task 的路径，方便用户执行其他操作。



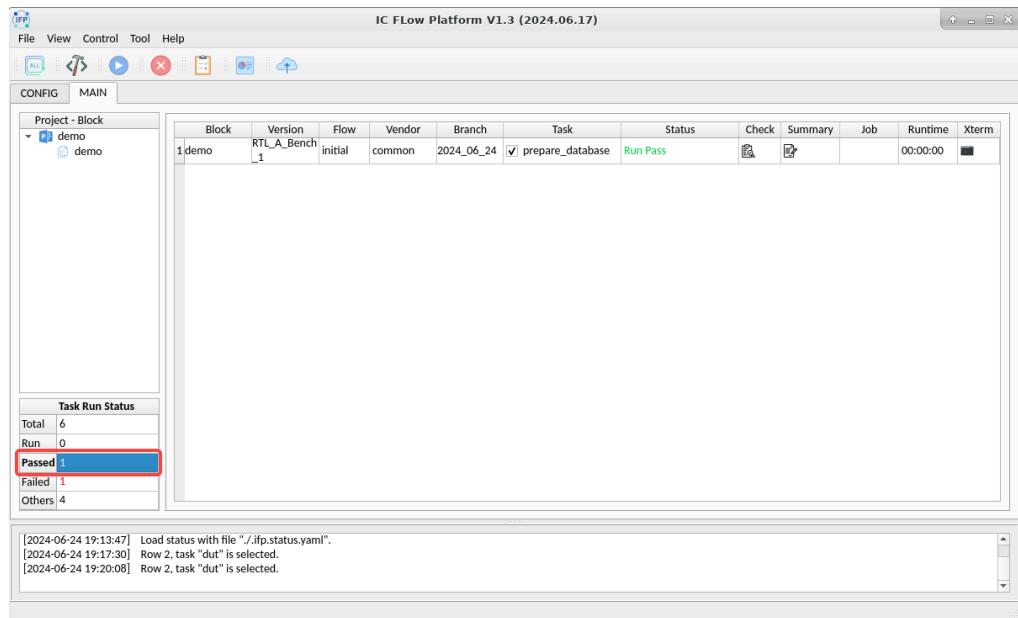
表头功能

右键点击 Task 栏位的表头，可通过 *Select Flows* 或 *Select Tasks* 功能批量选择 Task。



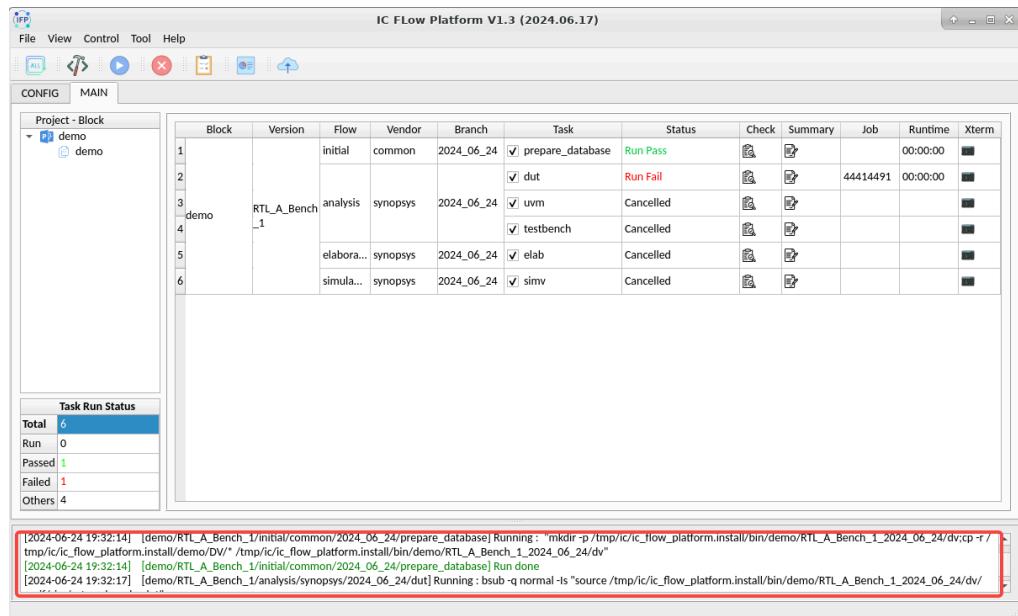
3.4.3 任务状态汇总区域

汇总了所有任务中 Run/Passed/Failed/Others 状态的任务数量，通过点击对应的栏位可筛选出该状态下所有的任务。

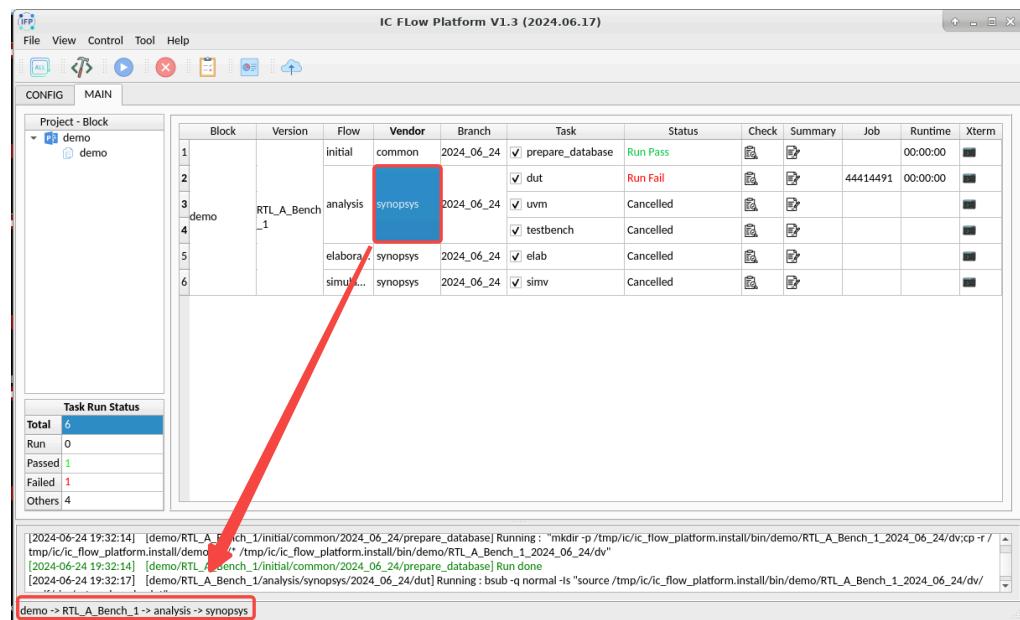


3.4.4 日志信息区域

展示运行信息，包括具体的任务层级、命令和完成状态。



当点击某个任务时，状态栏还将展示具体的层级信息。



4. 技术支持

本工具为开源工具，由开源社区维护，可以提供如下类型的技术支持：

- 部署和使用技术指导。
- 接收 bug 反馈并修复。
- 接收功能修改建议。(需审核和排期)

获取技术支持的方式包括：

- 通过 Contact 邮箱联系开发者。
- 添加 OPEN IC 小助手。



OPEN IC 小助手

4.1 变更历史

日期	版本	变更描述
2023.02.02	1.0	工具开源，第一个正式版本发布。
2023.07.14	1.1	修正部分操作逻辑 bug，优化 CONFIG 界面操作方式。
2023.08.31	1.1	优化菜单栏功能和界面操作方式。
2023.12.31	1.2	支持更为复杂的任务执行逻辑； 新增系统设置界面便于用户进行个性化设置。
2024.03.11	1.2.1	新增 dv 案例；

		新增可调用系统变量，优化任务信息展示。
2024.07.15	1.3	全新的用户配置界面； 新增 API 功能以适应不同用户的工作场景。

4.2 外部贡献者

感谢如下外部贡献者，为 IFP 提供了大量的 bug 反馈和代码优化建议，促进了 IFP 的快速迭代开发。

Soren Zhao

Xiaojiang